

P9070

LIFE • WORKS
FORMAT
LANGUAGE
MANUAL

Volume 2

Hackerspace Arnhem
Broekstraat 18
6828 PZ Arnhem
www.hack42.nl

#42
MADE IN
HACKERSPACE ARNHEM

PHILIPS



A publication of:

Philips Telecommunicatie en Data Systemen Nederland B.V.
SSS - Training & Documentation
P.O. Box 245
7300 AE Apeldoorn, The Netherlands

Printed in The Netherlands, April 1988

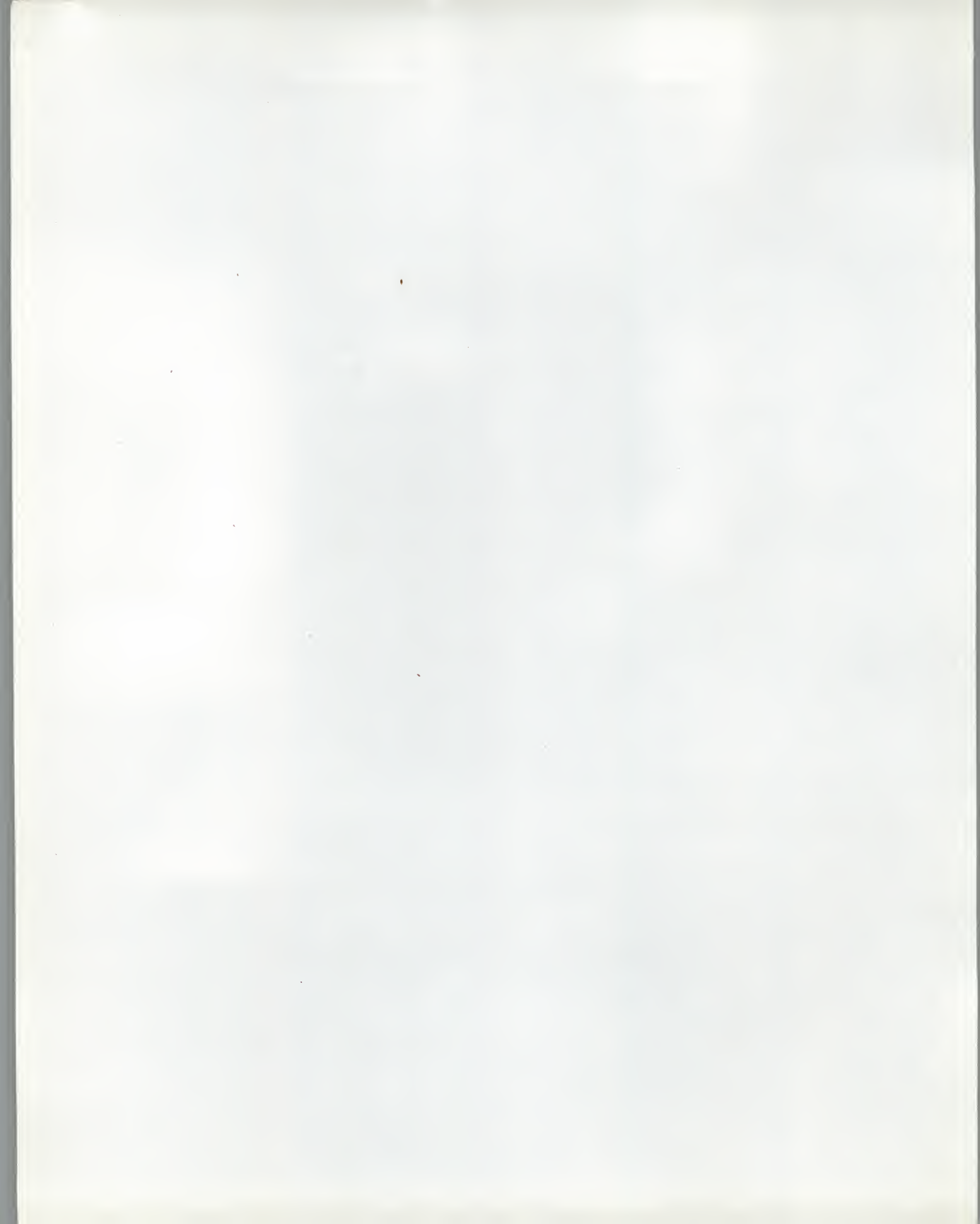
All rights are reserved. Reproduction in whole or in part is prohibited without the written consent of the copyright owner.

The information contained in this publication is accurate to the best of Philips' knowledge. However, Philips disclaims any liability resulting from the use of this information.

Order number: 5122 995 11332

Manual number: G312A





**LifE•Works Format Language Manual,
Volume 2**

List of Effective Pages

This publication contains 142 pages consisting of the following:

Page Number	Date	Page Number	Date
Front Cover.....	1 Nov 87	7-1 thru 7-11.....	1 Nov 87
Inside Front Cover (Blank) ..	1 Nov 87	7-12 (Blank)	1 Nov 87
Title	1 Nov 87	8-1 thru 8-11.....	1 Nov 87
A.....	1 Nov 87	8-12(Blank)	1 Nov 87
i.....	1 Nov 87	A-1 thru A-9.....	1 Nov 87
ii (Blank).....	1 Nov 87	A-10 (Blank).....	1 Nov 87
iii and iv	1 Nov 87	B-1 thru B-11	1 Nov 87
1-1 thru 1-4.....	1 Nov 87	B-12 (Blank)	1 Nov 87
2-1 thru 2-5.....	1 Nov 87	Index-1 thru Index-7	1 Nov 87
2-6 (Blank)	1 Nov 87	Index-8 (Blank)	1 Nov 87
3-1 thru 3-20.....	1 Nov 87	User's Comments	- - -
4-1 thru 4-32.....	1 Nov 87	Reply Card.....	- - -
5-1 thru 5-4.....	1 Nov 87	Inside Back Cover (Blank)...	- - -
6-1 thru 6-9.....	1 Nov 87	Back Cover	- - -
6-10 (Blank)	1 Nov 87		

* Asterisks indicate pages changed, added or deleted by the current change. The issue date for the change appears in place of the previous issue date at the bottom of each changed page included in the change package. Where a change involves a technical correction or the addition of new material, a vertical line appears at the appropriate place in the margin of the affected page. Deletions and editorial corrections are not specifically indicated.

Stock Number: 87601985A C

Issue A: 15 October 1986

Issue B: 15 February 1987

Issue C: 1 November 1987

Specifications subject to change.

Copyright ©1987

Motorola Inc.

All rights reserved.

Printed in U.S.A.

Preface

This manual explains how to use Multistation File Processing (MSFP) format coding elements and techniques in LiFE-Works format programs.

The manual is intended for experienced VISION and LiFE-Works format programmers who want to use MSFP. You must be familiar with format programming as described in the LiFE-Works Format Language Manual, Volume 1.

This manual describes release VA03 of LiFE-Works which is a new product evolving from VISION II. For MSFP users, LiFE-Works adds new async (AS) and operating system (UX) file types as well as printing through the Motorola Print System.

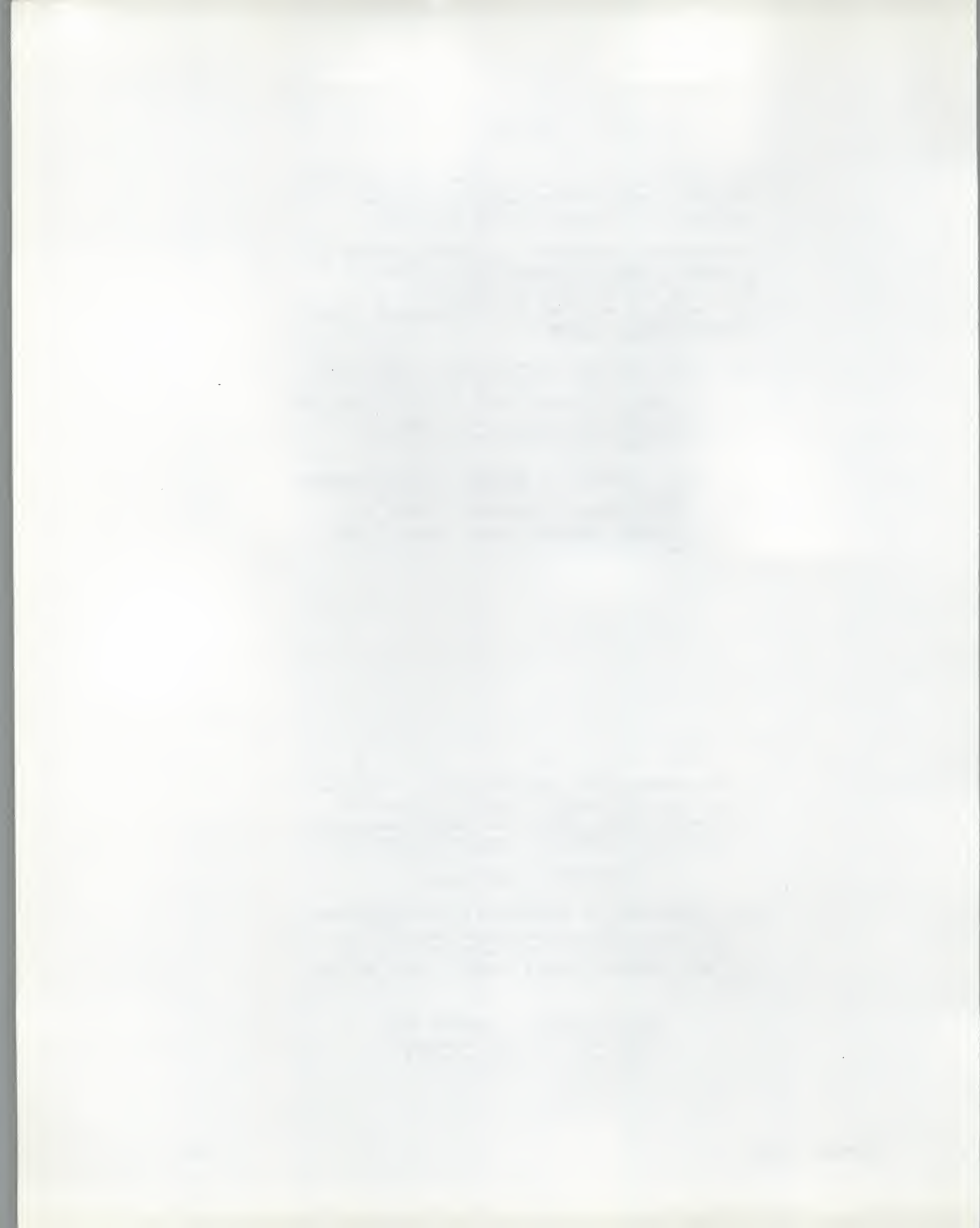
This manual replaces the VISION II Format Language Manual, Volume 2. Vertical bars in the outer margin mark technical differences between the VISION II Format Language Manual, Volume 2 and this manual.

If the documentation contained herein is supplied, directly or indirectly, to the U. S. Government, then the following notice shall apply unless agreed to in writing by Motorola Computer Systems, Inc.:

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in paragraph (b)(3)(ii) of the Rights in Technical Data and Computer Software clause at DFAR 252.227-7013.

Motorola Computer Systems, Inc.
10700 North De Anza Boulevard
Cupertino, California 95014



Contents

- 1 Introduction
 - What Is MSFP?, 1-1
 - Batch Processing and Transaction Processing, 1-1
 - MSFP's File-Handling Features, 1-2
 - About This Manual, 1-2
 - Summary of Contents, 1-3
 - Conventions and Terminology, 1-3
- 2 Basic Concepts
 - The MSFP Programming Elements, 2-1
 - The Processing Loop, 2-3
 - Programming Considerations, 2-4
 - Restrictions for Series 4000/5000 VISION, 2-5
- 3 Creating File Type Definitions
 - About File Type Definitions, 3-1
 - The \$FTYPE Supervisory Command, 3-1
 - The FTYPEfn Format Command, 3-2
 - Creating an SD (Sequential Disk) File Type Definition, 3-3
 - Creating an XD (Indexed Sequential Disk) File Type Definition, 3-6
 - Creating a PR (Printer) File Type Definition, 3-8
 - Creating an AS (Asynchronous) File Type Definition, 3-11
 - Creating a UX (Operating System) File Type Definition, 3-17
 - Generating a File Type Definition into the FTYPEfn Format Command, 3-19
- 4 Basic Commands
 - Introduction, 4-1
 - The CLOSEfn Command, 4-3
 - The DELETEfn Command, 4-5
 - The INSERTfn Command, 4-7
 - The LOCATEfn Command, 4-9
 - The OPENfn Command, 4-12
 - The READfn Command, 4-20
 - The RELEASEfn Command, 4-26
 - The REWRITEfn Command, 4-27
 - The WRITEfn Command, 4-29
- 5 Programming Elements Used with Record Buffers
 - Introduction, 5-1
 - The BLANKRfn Command, 5-2
 - The Rfn Validation/Generation Descriptor, 5-3

6 Conditional Indicators

- Introduction, 6-1
- The BOBfn Conditional Indicator, 6-2
- The BOFfn Conditional Indicator, 6-3
- The EOBfn Conditional Indicator, 6-4
- The EOFfn Conditional Indicator, 6-5
- The OPENEDfn Conditional Indicator, 6-6
- The OPENINfn Conditional Indicator, 6-7
- The OPENOUTfn Conditional Indicator, 6-8
- The RECSLfn Conditional Indicator, 6-9

7 Validation/Generation Descriptors

- Introduction, 7-1
- The #BATCHfn System Keyword, 7-2
- The #ERRORfn System Keyword, 7-3
- The #FILEfn System Keyword, 7-5
- The #JOBfn System Keyword, 7-6
- The #NODEfn System Keyword, 7-7
- The #PLEVELfn System Keyword, 7-8
- The #RECLFNfn System Keyword, 7-9
- The #RECNUMfn System Keyword, 7-11

8 MSFP Printing

- About MSFP Printing, 8-1
- Using a Vertical Format Unit (VFU), 8-2
- Defining MSFP Forms, 8-3
- Using Spool Files, 8-5
- Using the Spool File Display (Mode P-S), 8-7
 - Duplicate Spool File Name Selection Screen, 8-8
 - Using the File Detail Display, 8-10

A Sample Programs

- Example 1: Maintaining an Index Set, A-1
- Example 2: Reformatting Data as It Is Entered, A-6

B MSFP Printing with lp

- About MSFP Printing, B-1
- Creating a Vertical Format Unit (VFU) Definition, B-2
- Defining MSFP Forms, B-5
- Using Spool Files, B-7
- Using the Spool File Display (Mode P-S), B-10

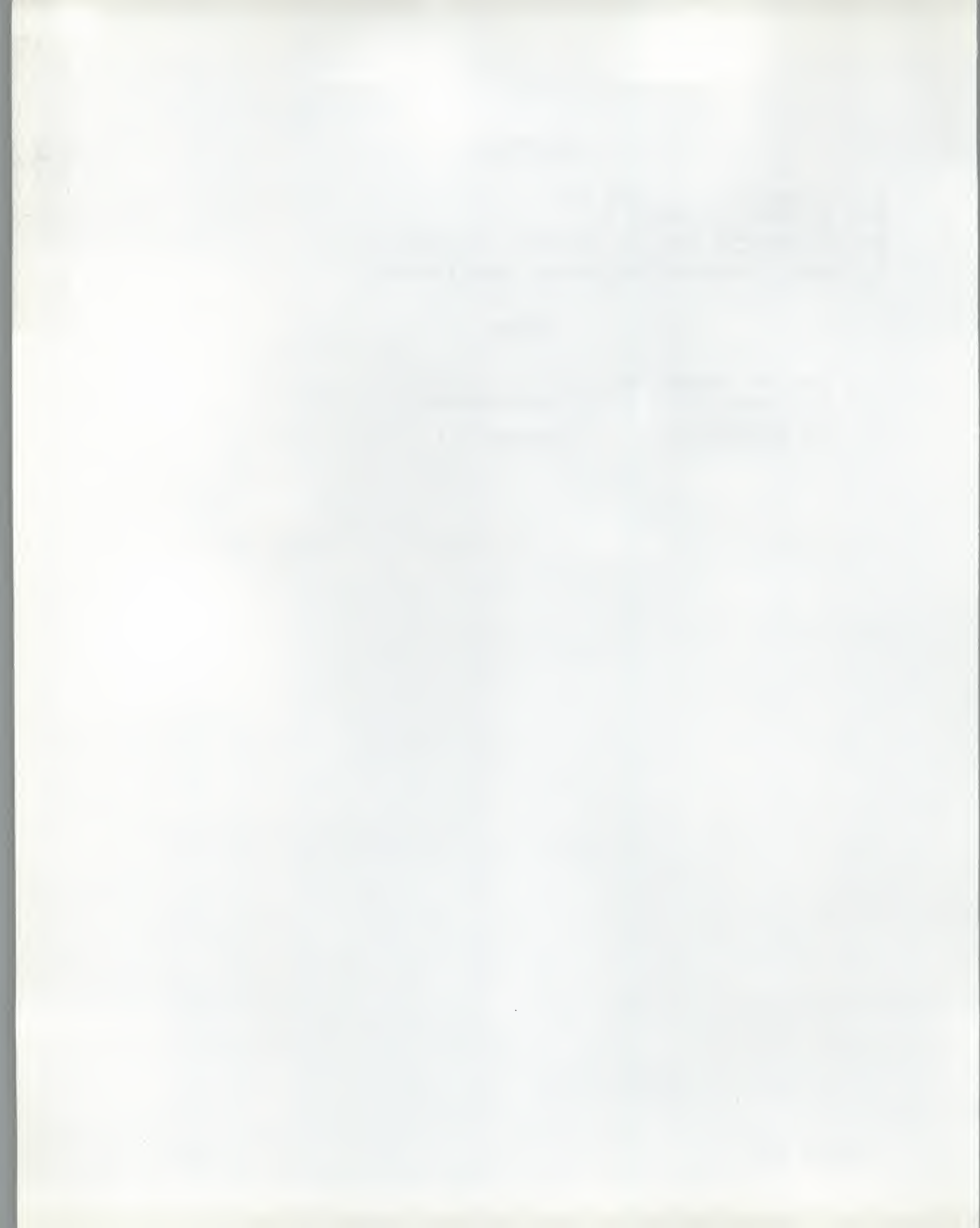
Index

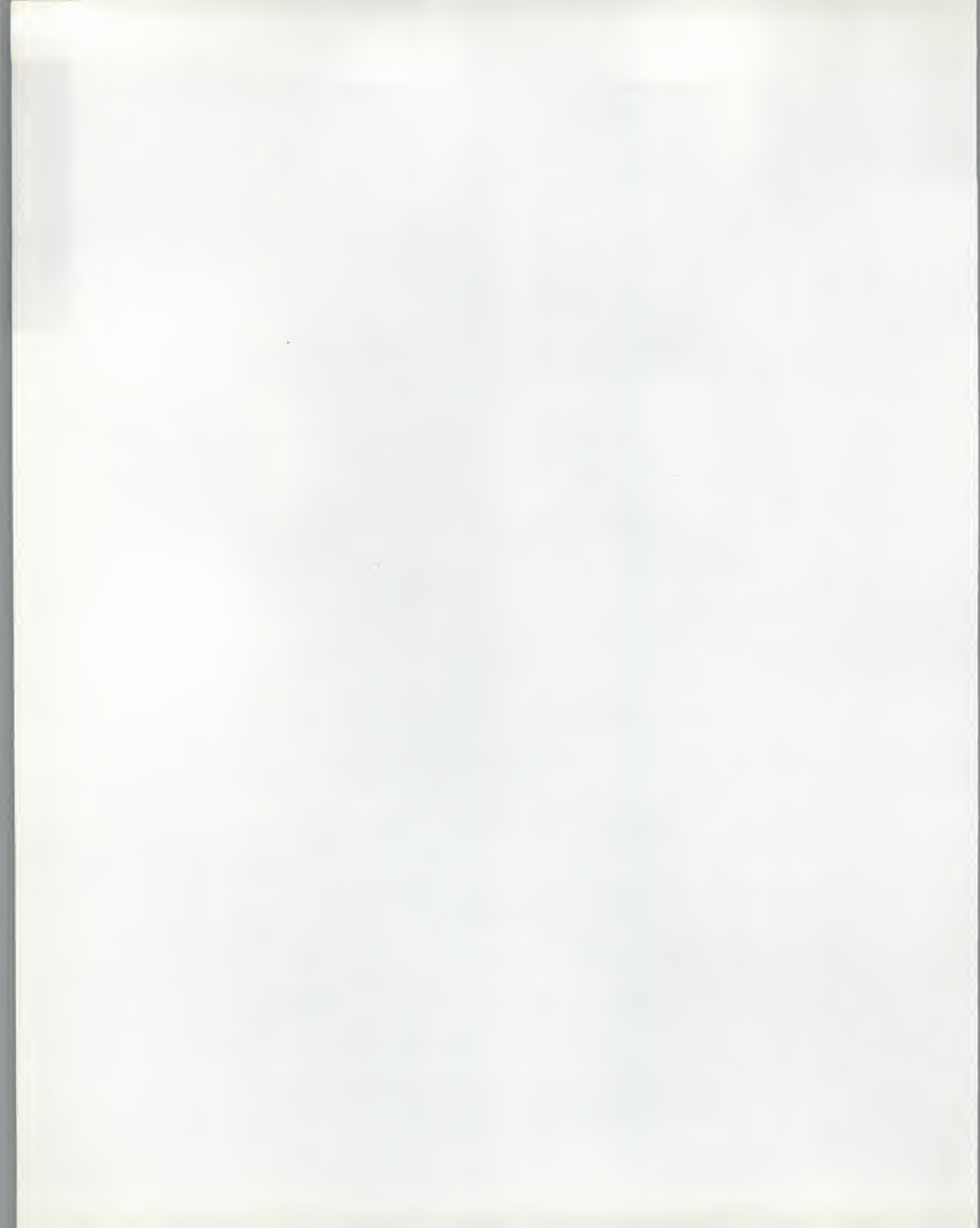
Illustrations

- 2-1 A Sample Processing Loop, 2-3
- 8-1 The SPOOL FILE DISPLAY, 8-7
- 8-2 The DUPLICATE SPOOL FILE NAME SELECTION Screen, 8-9
- 8-3 The FILE DETAIL Display, 8-10
- B-1 Mode P-S Display for an Individual Spool File, B-10

Tables

- 4-1 Basic MSFP Commands, 4-1
- 5-1 Programming Elements Used with Record Buffers, 5-1
- 6-1 MSFP Conditional Indicators, 6-1
- 7-1 MSFP Validation/Generation Descriptors, 7-1





Section 1 Introduction

WHAT IS MSFP?

MSFP (Multistation File Processing) is a set of specialized supervisory commands and format coding elements. MSFP extends and enhances the kinds of file processing you can do through LIfE-Works format code.

MSFP adds two major capabilities to LIfE-Works format programming: it lets you do transaction processing rather than batch processing, and it lets you write programs that are independent of the files or physical devices being manipulated. Each of these capabilities is described below.

MSFP is an extension of the LIfE-Works format language, not a separate language. An "MSFP program" contains both MSFP commands and regular format commands. This manual assumes that you are familiar with LIfE-Works format programming as described in the LIfE-Works Format Language Manual, Volume 1.

Batch Processing and Transaction Processing

Most LIfE-Works file operations are batch-oriented. In batch processing, a large amount of data is collected, then the data is manipulated all at once. For example, an operator might enter data into a batch during the day, then reformat and print the batch at the end of the day.

Batch processing is an efficient way to deal with whole batches of data. However, batch processing implies a certain amount of delay between the time data is entered and the time it is processed. In addition, a batch is the smallest unit handled. Batch processing isn't suited to time-critical tasks or tasks that involve only a few records of a batch.

MSFP is transaction-oriented. In transaction processing, data can be manipulated as soon as it is entered, and immediate results can be produced on a record-by-record basis. Transaction processing is suitable for tasks like these:

- Printing a receipt for a purchase.
- Making an airline reservation.
- Reissuing a miscalculated paycheck.
- Updating a bank customer's account balance.

Some tasks may require a mixture of batch processing and transaction processing. For example, you might use transaction processing to update records during the day, then use batch processing to print a summary of the day's activity.

NOTE

Like MSFP, LIfE-Works index sets are transaction-oriented. Many index set operations can be done through either MSFP or non-MSFP methods. MSFP lets you do some index set operations that aren't possible with regular LIfE-Works commands.

MSFP's File-Handling Features

MSFP lets you write format programs that are independent of the data file or device being manipulated. MSFP also makes it easier to write code for tasks involving multiple files or devices.

Under MSFP, the concept of a "file" is broadened to mean anything MSFP can write to and/or read from. MSFP recognizes the following types of files:

- LIfE-Works batches (called sequential disk or SD files)
- LIfE-Works index sets (called indexed sequential disk or XD files)
- Printers (called PR files)
- Asynchronous devices (called AS files)
- Operating system files (called UX files)

In general, you use the same commands to manipulate any MSFP file. This gives you a unified, standard way to handle different types of files or devices. You can manipulate up to 16 files (in any combination of types) in a single MSFP program.

For example, you can write a program that takes data from the screen and reformats the data to a target file. If you specify a LIfE-Works batch as the target file, the program stores the data in the batch. If you specify a printer as the target file, the same program prints the data.

Note that you don't have to identify actual files or devices when you write the code; you can do that at any time prior to running the program.

Some MSFP commands apply to specific types of files. For example, there are commands that apply only to printers. In general, MSFP makes programs "interchangeable" by ignoring any command that doesn't apply to the type of file currently being processed. The only exceptions occur when an entire operation doesn't make sense for a given file type. For example, MSFP won't let you run a program that tries to read data from a printer.

ABOUT THIS MANUAL

This subsection describes the contents of this manual, explains the conventions used in command strings, and defines some commonly used terms.

Summary of Contents

This manual is organized as follows:

- Section 2, "Basic Concepts," contains a general overview of MSFP commands and procedures.
- Sections 3 through 8 describe the MSFP programming elements:
 - Section 3, "Creating File Type Definitions," describes commands that define a file or device so MSFP can access it.
 - Section 4, "Basic Commands," describes commands that manipulate files and records.
 - Section 5, "Programming Elements Used with Record Buffers," describes elements that manipulate data in record buffers.
 - Section 6, "Conditional Indicators," describes a set of conditional indicators used in MSFP programs.
 - Section 7, "Validation/Generation Descriptors," describes a set of system keywords used in MSFP programs.
 - Section 8, "MSFP Printing," describes MSFP printing using the Motorola Print System.
- Appendix A, "Sample Programs," contains two annotated examples of MSFP programs.
- Appendix B, "MSFP Printing with lp," describes MSFP printing through the operating system facility lp.

Conventions and Terminology

This manual uses the following conventions:

- Optional commands or parameters are enclosed in [brackets]. For example:

READfn,[R]

means that the R parameter of the READfn command is optional.

- Alternative choices are enclosed in {braces}. Choose one of the alternatives shown. For example,

LOCATEfn { $\begin{matrix} \text{BOF} \\ \text{EOF} \\ \text{expression} \end{matrix}$ }

Section 1
Introduction

means that you must use either the BOF parameter, the EOF parameter, or an arithmetic expression with the LOCATEfn command.

- The variable fn appears consistently in MSFP format programming elements. (Examples: OPENfn, #RECNUMfn.) fn represents a logical file number--a shorthand way of referring to a file in a format program. For example, the commands CLOSE2, LOCATE2, and READ2 all refer to logical file 2. fn can be a number from 1 to 16. This manual does not define fn every time it appears.
- The term record pointer means your current position within the file being processed. The record pointer can be at beginning-of-file (before the first record), on a record, or at end-of-file (at the end of the last record).



Section 2 Basic Concepts

THE MSFP PROGRAMMING ELEMENTS

This subsection describes the MSFP programming elements in general terms. Individual elements are described in detail in Sections 3 through 7 of this manual.

MSFP elements can be divided into the following categories:

- Commands that create file type definitions. A file type definition describes a physical file or device so MSFP can access it.
- Commands for manipulating files and records. You can:
 - Open a file.
 - Close a file.
 - Read a record from a file into a buffer.
 - Write a record from a buffer into a file.
 - Move the record pointer to a specified location within a file.
- Commands for manipulating data in record buffers. You can:
 - Access data in a record buffer for validation, generation, or updating.
 - Fill a record buffer with blanks.
- Conditional indicators, which are used with IF...THEN...ELSE statements. You can test for the following conditions:
 - Beginning of batch.
 - Beginning of file.
 - End of batch.
 - End of file.
 - File is open.
 - File is open for input.
 - File is open for output.
 - Record is selected from an index set.

Section 2
Basic Concepts

- Validation/generation descriptors. You can validate or generate the following values:
 - Current batch identifier
 - Current spool file name or operating system or asynchronous file name
 - Current job name
 - Current program level
 - Current record length
 - Current record number
 - Current node name (OfficeLAN remote system name)
 - Current OfficeLAN error status
- Commands for manipulating index sets. You can:
 - Delete a record from an index set.
 - Insert a record into an index set.
 - Select and read an index set record that has a specified key field value.
 - Select and read an index set record that has a key field value equal to or greater than a specified value.
 - Select and read index set records sequentially.
 - Release all access to an index set.

THE PROCESSING LOOP

Most MSFP programs contain a basic processing loop:

1. Open the files.
2. Process the files.
3. Close the files.

Figure 2-1 illustrates a typical processing loop.

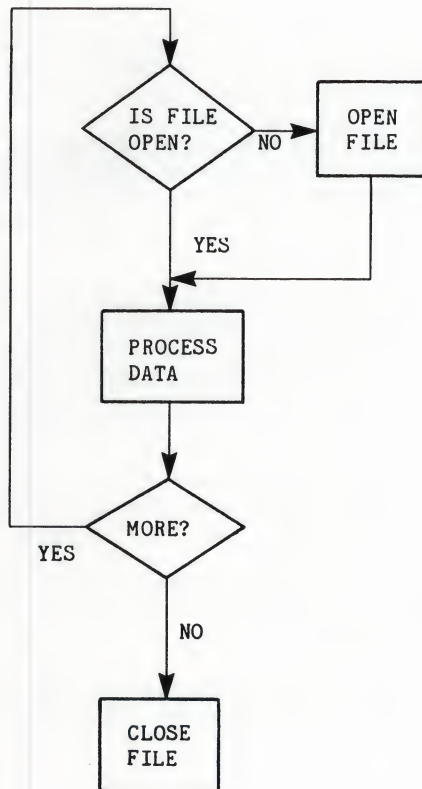


Figure 2-1. A Sample Processing Loop

PROGRAMMING CONSIDERATIONS

- Before you can do any processing in an MSFP file, you must use the OPENfn command to open the file. You should close each file (using the CLOSEfn command) as soon as your program no longer needs access to the file.
- All data manipulation under MSFP is done in buffers. Two kinds of buffers are available to MSFP: work buffers and record buffers.

Both MSFP and non-MSFP programs can use work buffers. You can allocate one work buffer per terminal by using the GETBUF format command. For more information on work buffers, see the LIFe-Works Format Language Manual, Volume 1.

Only MSFP programs can use record buffers. LIFe-Works automatically allocates one record buffer for each file opened in an MSFP program. Each record buffer corresponds to a specific file. A record buffer's length is equal to the maximum record length defined for the corresponding file.

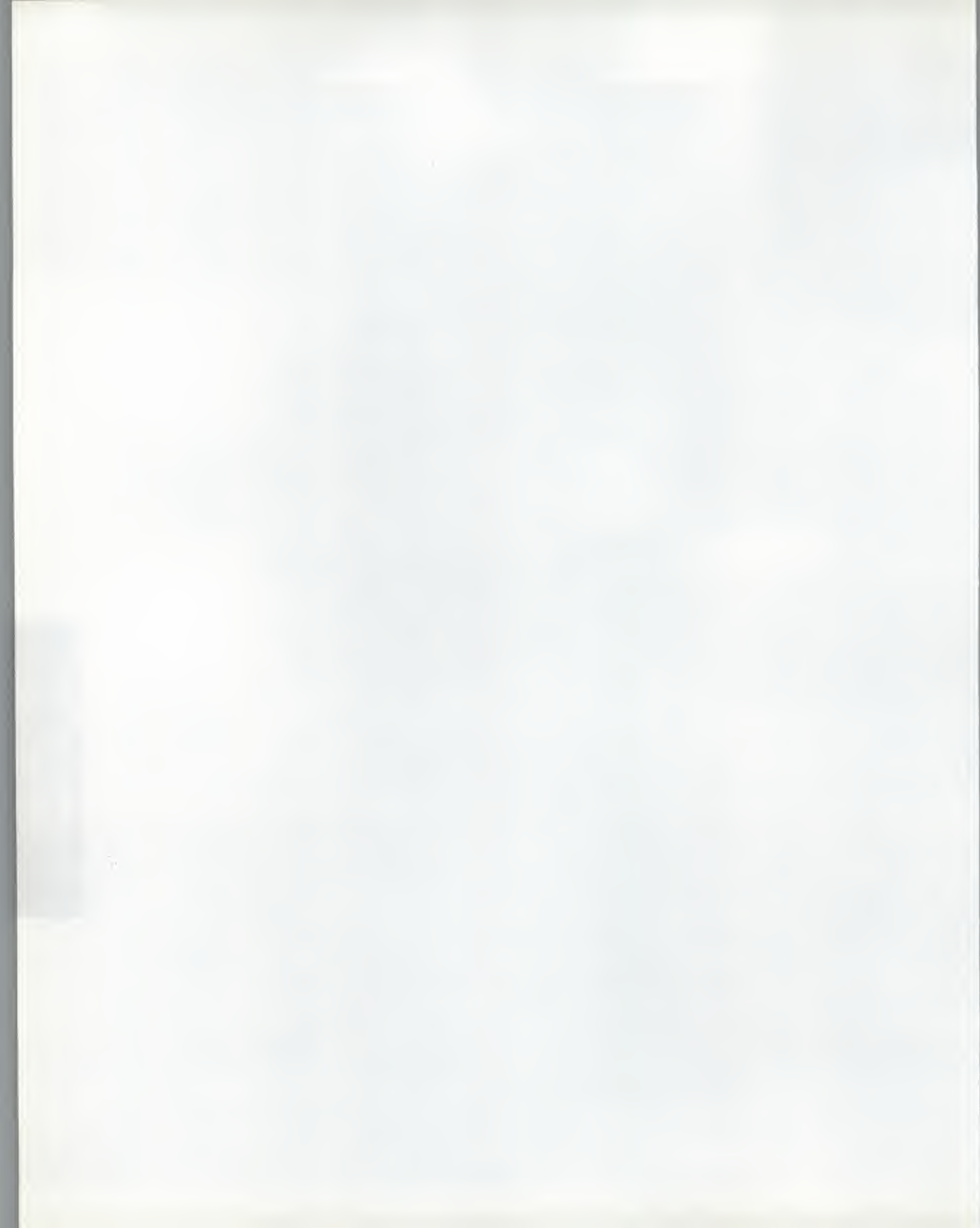
To process data in any way, you must read the data out of a file into a buffer. You can then update the data, validate against it, or generate from it. You can also transfer data from one buffer to another. When you finish processing the data, you can store it by writing it out of a buffer into a file.

- In addition to the 16 possible MSFP files, each MSFP program also contains a job and batch that supplies the operator interface. This job and batch is not an MSFP file; its only purpose is to prompt the operator to start processing, enter information at decision points, and so on. The operator's entries are often not written to disk (NDR). See Appendix A, "Sample Programs," for examples that illustrate this concept.
- As you write your code, keep in mind that there can be problems if the operator backs up and changes data that MSFP has already processed. Two tips for avoiding these problems:
 - Use the NOBACK and NOUP format commands to prevent the operator from backing up. NOBACK and NOUP are described in the LIFe-Works Format Language Manual, Volume 1.
 - Code that writes data out from a buffer to a file should be placed at the end of the format, so it executes after the operator has entered the last field.

RESTRICTIONS FOR SERIES 4000/5000 VISION

If your Series 8000 Life-Works system is connected to a Series 4000/5000 VISION system on an OfficeLAN network, certain restrictions apply to MSFP file handling on the Series 4000/5000 system:

- Only SD and XD file types are permitted. PR, AS, and UX file types are not supported.
- \$ORDER chains are not supported.
- For XD file types, #JOB and #BATCH are set to blanks when an index set is opened with the OPENfn command.



Section 3 Creating File Type Definitions

ABOUT FILE TYPE DEFINITIONS

This section explains how to create file type definitions for each kind of MSFP file.

A file type definition is a way of identifying a physical file or device to MSFP. You must create a file type definition for each file or device you want to access through an MSFP program. The MSFP file types and the corresponding files and devices are:

<u>File type</u>	<u>Physical file or device</u>
SD	LiFE-Works batch (sequential disk file)
XD	LiFE-Works index set (indexed sequential disk file)
PR	Printer
AS	Asynchronous device
UX	Operating system file

There are two ways to create a file type definition: the \$FTYPE supervisory command and the FTYPEfn format command.

Note that you can also use the LiFE-Works menu system to define file types. This method makes use of the "Add/Delete I/O File" menu, and performs the same function as the \$FTYPE supervisory command. For further information, see the LiFE-Works Menu Manual.

The \$FTYPE Supervisory Command

The \$FTYPE supervisory command lets you create a file type definition that is independent of format code. File type definitions created by using \$FTYPE are stored on disk as system elements, much like formats or index sets. These stored file type definitions can be used by any number of MSFP programs.

Use \$FTYPE to create file type definitions that can be used repeatedly. For example, you can create a file type definition that identifies a frequently used printer/form combination. \$FTYPE lets you create up to 1000 file type definitions, identified by the numbers 000-999.

Section 3
Creating File Type Definitions

NOTE

The term Ftype is used as an abbreviation for "file type definition" when referring to a stored system element. (Examples: Ftype 100, Ftype 222.) You can get information about the Ftypes on your system through mode S-T and mode S-U. Mode S-T lists existing Ftypes and provides detailed information on each Ftype. Mode S-U lists Ftypes by type (SD, PR, and so on). The mode S-T and mode S-U displays are described in the LifE-Works Supervisor's Manual.

The FTYPEfn Format Command

The FTYPEfn format command lets you create a file type definition within a format program. File type definitions created by using FTYPEfn are not stored as system elements; they exist only while the program is being executed. They cannot be used by other programs and are not listed anywhere on the system.

Use FTYPEfn when you don't need to store a file type definition for reuse. The FTYPEfn format command uses fewer system resources than the \$FTYPE supervisory command, and gives you an unlimited number of file type definitions.

CREATING AN SD (SEQUENTIAL DISK) FILE TYPE DEFINITION

A sequential disk file can be a single LiFE-Works batch or a group of batches identified in an order chain. The abbreviation SD identifies a sequential disk file.

To create an SD file type definition using the \$FTYPE supervisory command, enter and execute:

```
$FTYPE nnn,SD $JOB job { ,batch  
                        ,*  
                        $ORDER } [$NODE node]
```

where

nnn is a three-digit number from 000 to 999 assigned as the identifier to the specified file type definition.

job identifies the job name that contains a batch or batches to be manipulated.

batch is the batch name.

* indicates that the batch name is to be entered when the file is opened.

\$ORDER indicates that the SD file consists of batches specified in a previously executed \$ORDER command for the job job.

node identifies the remote system where this file resides. node is a system name from 1 to 20 characters long. If your system uses OfficeLAN, you can manipulate files on the other systems ("nodes") in your LAN network.

To create an SD file type definition using the FTYPEfn format command, enter:

```
FTYPEfn = "SD $JOB job { ,batch  
                        ,*  
                        $ORDER } [$NODE node]"
```

where

fn is a number from 1 to 16 assigned as the logical file number to the specified file type definition.

job is the job name that contains a batch or batches to be manipulated.

batch is the batch name.

Section 3

Creating File Type Definitions

* indicates that the batch name is to be entered when the file is opened.

\$ORDER indicates that the SD file consists of batches specified in a previously executed \$ORDER command for the job job.

node identifies the remote system where this file resides. node is a system name from 1 to 20 characters long. If your system uses OfficeLAN, you can manipulate files on the other systems ("nodes") in your LAN network.

Usage Notes

- Instead of supplying a literal string on the right side of the FTYPEfn format command, you can generate the right side of the command from another source, such as a field or buffer. For details, see "Generating a File Type Definition into the FTYPEfn Format Command" at the end of this section.
- The \$ORDER parameter lets you define multiple batches of a job as a single SD file. Note that MSFP uses only the batches that are named in the \$ORDER command, not every batch in the job. Note also that purging, clearing, or linking any batch in a job deactivates the order chain for that job.
- If you use a literal string rather than a generated value on the right side of the FTYPEfn command, you must enclose the entire right side in single or double quotation marks.
- If the sequential disk file resides on your own system, do not use the \$NODE parameter. If the file resides on another system on the OfficeLAN network, you must use the \$NODE parameter.

Examples

```
$FTYPE 001,SD $JOB PAYROLL,1
```

Assigns file type identifier 001 to batch 1 of the job PAYROLL.

```
FTYPE,4 = "SD $JOB INVENTORY,*"
```

Assigns logical file number 4 to a batch of the job INVENTORY. The batch name is requested at run time.

\$ORDER CONTRACTS,DIST1,DIST2,DIST3

\$FTYPE 350,SD \$JOB CONTRACTS \$ORDER

Assigns file type identifier 350 to batches DIST1, DIST2, and DIST3 of the job CONTRACTS. The batches have been identified in a previously executed \$ORDER command.

\$FYTPE 001,SD \$JOB PAYROLL,1 \$NODE ACCOUNTING

Assigns file type identifier 001 to batch 1 of the job PAYROLL, which resides on a system named ACCOUNTING.

Section 3

Creating File Type Definitions

CREATING AN XD (INDEXED SEQUENTIAL DISK) FILE TYPE DEFINITION

An indexed sequential disk file is a LiFE-Works index set. The abbreviation XD identifies an indexed sequential disk file.

To create an XD file type definition using the \$FTYPE supervisory command, enter and execute:

```
$FTYPE nnn,XD [$INDSET id] [$NODE node]
```

where

nnn is the identifier assigned to the specified file type definition. If you include the optional \$INDSET parameter, nnn can be any three-digit number from 000 to 999. If you don't use \$INDSET, nnn must be the same as the three-digit identifier for the index set being manipulated.

id identifies the index set to be manipulated. Optional for \$FTYPE.

node identifies the remote system where this file resides. node is a system name from 1 to 20 characters long. If your system uses OfficeLAN, you can manipulate files on the other systems ("nodes") in your LAN network.

To create an XD file type definition using the FTYPEfn format command, enter:

```
FTYPEfn = "XD $INDSET id [$NODE node]"
```

where

fn is a number from 1 to 16 assigned as the logical file number to the specified file type definition.

id identifies the index set to be manipulated. id is required.

node identifies the remote system where this file resides. node is a system name from 1 to 20 characters long. If your system uses OfficeLAN, you can manipulate files on the other systems ("nodes") in your LAN network.

Usage Notes

- Instead of supplying a literal string on the right side of the FTYPEfn format command, you can generate the right side of the command from another source, such as a field or buffer. For details, see "Generating a File Type Definition into the FTYPEfn Format Command" at the end of this section.

- If you use a literal string rather than a generated value on the right side of the FTYPEfn command, you must enclose the entire right side in single or double quotation marks.
- If the indexed sequential disk file resides on your own system, do not use the \$NODE parameter. If the file resides on another system on the OfficeLAN network, you must use the \$NODE parameter.

Examples

```
$FTYPE 100,XD
```

Assigns file type identifier 100 to index set 100.

```
$FTYPE 100,XD $INDSET 200
```

Assigns file type identifier 100 to index set 200.

```
FTYPE3 = "XD $INDSET 200"
```

Assigns logical file number 3 to index set 200.

```
FTYPE16 = "XD $INDSET 300 $NODE LIBRARY"
```

Assigns logical file number 16 to index set 300, which resides on a system named LIBRARY.

Section 3
Creating File Type Definitions

CREATING A PR (PRINTER) FILE TYPE DEFINITION

The abbreviation PR identifies a printer file.

To create a PR file type definition using the \$FTYPE supervisory command, enter and execute:

```
$FTYPE nnn,PR [$FORMS { vvv } ] [$LENGTH lll] [$SPOOL { spoolfile } ]  
                [ $PRINTER printer-name]
```

where

nnn is assigned as the identifier to the specified file type definition.

vvv or * identifies the MSFP form to be used. vvv is the form identifier. If you use an asterisk instead of specifying a form identifier, the identifier is requested at run time. If you omit \$FORMS, MSFP prints on the printer and form currently assigned to the terminal running the program.

lll defines the length of the print line. lll is the number of characters per line (264 maximum). If you omit \$LENGTH, you get whatever line length was specified in the V2config form. If no line length was specified in the V2config form, you get 132 characters per line. All print lines are fixed-length.

spoolfile or * assigns a name to a temporary or permanent spool file. spoolfile is a spool file name (up to 14 characters). If you use an asterisk instead of specifying a name, the spool file name is requested at run time.

printer-name is the name of the printer to which output should spool. This name overrides the name of the printer the terminal is associated with via the \$SETPRINT command. If \$PRINTER is not specified, the default printer is used.

To create a PR file type definition using the FTYPEfn format command, enter:

```
FTYPEfn = "PR [$FORMS { vvv } ] [$LENGTH lll] [$SPOOL { spoolfile } ]"  
          [ $PRINTER printer name]
```

where

fn is a logical file number from 1 to 16 assigned to the specified file type definition.

vvv or * identifies the MSFP form to be used. vvv is the form identifier. If you use an asterisk instead of specifying a form identifier, the identifier is requested at run time. If you omit \$FORMS, MSFP prints on the printer and form currently assigned to the terminal running the program.

lll defines the length of the print line. lll is the number of characters per line (264 maximum). If you omit \$LENGTH, you get whatever line length was specified in the V2config form. If no line length was specified in the V2config form, you get 132 characters per line. All print lines are fixed-length.

spoolfile or * assigns a name to a temporary or permanent spool file. spoolfile is a spool file name (up to 14 characters). If you use an asterisk instead of specifying a name, the spool file name is requested at run time.

printer-name is the name of the printer to which output should spool. This name overrides the name of the printer the terminal is associated with via the \$SETPRINT command. If \$PRINTER is not specified, the default printer is used.

Usage Notes

- Instead of supplying a literal string on the right side of the FTYPEfn format command, you can generate the right side of the command from another source, such as a field or buffer. For details, see "Generating a File Type Definition into the FTYPEfn Format Command" at the end of this section.
- See Section 8 for more information on MSFP forms, spool files, and other topics related to printing.
- If you use a literal string rather than a generated value on the right side of the FTYPEfn command, you must enclose the entire right side in single or double quotation marks as shown above.
- If the printer resides on another system on the OfficeLAN network, just specify the name of the printer. You don't need to use the \$NODE parameter. OfficeLAN will find the correct node for whatever printer you specify in the \$FORMS definition. No matter where the printer resides, if you identify a spool file it will reside on your system.

Examples

```
$FTYPE 500,PR $FORMS 400
```

Assigns the file type identifier 500 to the printer/form combination identified in MSFP form definition 400.

Section 3
Creating File Type Definitions

FTYPE1 = "PR \$FORMS 100 \$LENGTH 80 \$SPOOL REPORT \$PRINTER LPR-1"

Assigns logical file number 1 to the printer/form combination identified in MSFP form definition 100. Specifies that the print line is 80 columns wide, the spool file is to be named REPORT, and output is to go to printer LPR-1.

CREATING AN AS (ASYNCHRONOUS) FILE TYPE DEFINITION

The abbreviation AS identifies an asynchronous file.

To create an AS file type definition using the \$FTYPE supervisory command, enter and execute:

```
$FTYPE nnn,AS $PATH path [$STTY modes] [$TIMEOUT sec] [$WAIT] [$MAXSIZE lll]  
[$EORM \hh]
```

where

nnn is a three-digit number from 000 to 999 assigned as the identifier to the specified file type definition.

path is the operating system path name to the device to be opened as this file type. The last component of the path name may be an asterisk (*), indicating that it will be requested when the file is opened. The path name must be less than 80 characters.

modes is any number or combination of mode settings allowed for stty(1), an operating system utility. (See the System V/68 User's Manual for details.) You can also specify "alloff" to turn off all \$STTY mode settings. Any modes not specified with \$STTY default to the "sane" settings. Note that \$STTY allows two mode settings, "idevice" and "odevice," not normally supported by the stty(1) utility. See "Usage Notes" below for further information.

sec is the number of seconds, up to a maximum of 999999, that can pass before an OPENfn, READfn, or WRITEfn command is interrupted. A value of 0 means "do not timeout." If \$TIMEOUT is not specified, the default value is 45 seconds.

\$WAIT indicates that the OPENfn command is to wait for a physical connection to be made--that is, wait for DTR signal. If \$WAIT is not specified, the default is do not wait.

lll is the record buffer size, up to 9999 characters. If \$MAXSIZE is not specified, the default value is 1500 characters.

\hh is the hexadecimal representation of a character to be used as an end-of-record marker. This character is appended to the end of a record before it is written. The end-of-record character is not included in the record length. If \$EORM is not specified, no end-of-record character is appended.

To create an AS file type definition using the FTYPEfn format command, enter:

```
FTYPEfn = "AS $PATH path [$STTY modes] [$TIMEOUT sec] [$WAIT] [$MAXSIZE lll]  
[$EORM \hh]"
```


Section 3

Creating File Type Definitions

where

fn is a number from 1 to 16 assigned as the logical file number to the specified file type definition.

path is the operating system path name to the device to be opened as this file type. The last component of the path name may be an asterisk (*), indicating that the device will be requested when the file is opened. The path name must be less than 80 characters.

modes is any number or combination of mode settings allowed for stty(1), an operating system utility. (See the System V/68 User's Manual for details.) You can also specify "alloff" to turn off all \$STTY mode settings. Any modes not specified with \$STTY default to the "sane" settings. Note that \$STTY allows two mode settings, "idevice" and "odevice," not normally supported by the stty(1) utility. See Usage Notes below for further information.

sec is the number of seconds, up to a maximum of 999999, that can pass before an OPENfn, READfn, or WRITEfn command is interrupted. A value of 0 means "do not timeout." If \$TIMEOUT is not specified, the default value is 45 seconds.

\$WAIT indicates that the OPENfn command is to wait for a physical connection to be made--that is, wait for DTR signal. If \$WAIT is not specified, the default is do not wait.

l11 is the record buffer size, up to 9999 characters. If \$MAXSIZE is not specified, the default value is 1500 characters.

\hh is the hexadecimal representation of a character to be used as an end-of-record marker. This character is appended to the end of a record before it is written. The end-of-record character is not included in the record length. If \$EORM is not specified, no end-of-record character is appended.

Usage Notes

- Instead of supplying a literal string on the right side of the FTYPEfn format command, you can generate the right side of the command from another source, such as a field or buffer. For details, see "Generating a File Type Definition into the FTYPEfn Format Command" at the end of this section.
- If you use a literal string rather than a generated value on the right side of the FTYPEfn command, you must enclose the entire right in single or double quotes as shown above.
- The "TO sec" option in the OPENfn command can be used to override the \$TIMEOUT value specified in an AS ftype definition. See "The OPENfn Command" in Section 4 for details.
- You can use the UNTIL= option of the READfn command to check for an end-of-record character specified with the \$EORM option in \$FTYPE command.

- The "alloff" \$STTY mode setting turns off all \$STTY mode settings and sets all control characters to undefined. It is convenient to use "alloff" when you want to set just a few modes or control characters. See "Examples" below.
- The "idevice" and "odevice" \$STTY mode settings assign a combination of mode settings for input and output devices, respectively. Use "idevice" for quick setup of an AS file type that reads input from an alternate keyboard device, such as a bar code reader. Use "odevice" for quick setup of an AS file type that will write directly to a printer.

The "idevice" mode sets the port to reasonable values for devices like bar code readers and scanners. These values differ from "sane" mode settings in that flow control is turned on (enable xon/xoff protocol) and echoing is turned off (disable all modes relating to character echo).

Similarly, "odevice" mode sets the port to reasonable values for devices like printers. These values were copied from the 9600 baud printer setup specified in the /usr/spool/lp/model/ptnnx file. As stated in the ptnnx file, the rules for 9600 baud printers are 8-bit characters, no parity, and one stop bit.

- Notes on operating system prerequisites for AS file usage:
 - Insure that the port to be used by the AS file is not active--that is, that the program "getty" is not active on the port. To display or modify the characteristics of a tty, see "tty management procedures" in the System V/68 System Administrator's Guide. For additional details, see getty(1M) and init(1M) in the System V/68 System Administrator's Reference Manual.
 - Insure that the device (also referred to as the port or the file in the /dev directory) has the correct permissions. Generally, if the port has been active (getty has been running), the permissions will be write-only (that is, 222 or c-w--w--w-). You must change the permissions to 666 or crw-rw-rw- if you intend to use the port for reading. See chmod(1) and ls(1) in the System V/68 User's Reference Manual for additional information.
 - You may need to set special I/O parameters for your asynchronous device. Refer to termio(7) in the System V/68 System Administrator's Reference Manual and stty(1) in the System V/68 User's Reference Manual. Pay particular attention to the discussion of the "icanon" mode under local modes in termio(7). Note that the EOF and EOL special characters become MIN and TIME respectively when canonical processing is disabled ("-icanon"). The setting of these values allows you to control input buffering.

Examples

```
$FTYPE 505,AS $PATH /dev/tty11 $MAXSIZE 2048
```

Assigns file type identifier 505 to the asynchronous device at path /dev/tty11. Sets the I/O parameters as if an "stty sane" command had been executed, and uses the default value of 45 seconds for timeout. Sets the record buffer size to 2048 characters. Since \$WAIT is not specified, the OPENfn command will not wait for DTR. No end-of-record character is written since \$EORM is not specified.

```
$FTYPE 505,AS $PATH /dev/tty11 $STTY -ignbrk brkint ignpar -parmrk -inpck  
istrip -inlcr -igncr icrnl -iuclic ixon ixany -ixoff opost -olcuc  
onclcr -ocrncl -onocr -onlret -ofill -ofdel -parenb -parodd cs8  
-ostopb hupcl cread -clocal b9600 isig icanon -xcase echo echoe  
echok -echonl -noflsh $MAXSIZE 2048
```

Performs exactly the same function as the command in the previous example. In this example, the parameter values for the "stty sane" command are explicitly set and unset.

```
$FTYPE 505,AS $PATH /dev/tty12 $STTY alloff brkint intr \7F
```

Assigns file type identifier 505 to the asynchronous device at path /dev/tty12. Turns off all stty settings and sets control characters to undefined, then sets brkint and defines the INTR control character as hexadecimal 7F. Uses the default values for \$TIMEOUT and \$MAXSIZE. No waiting on OPEN and no end-of-record characters are specified.

```
FTYPE3 = "AS $PATH /dev/* $STTY B1200 $TIMEOUT 20 $EORM \0D"
```

Assigns logical file number 3 to the asynchronous device. The last component of the path name (the device) is to be supplied when the file is opened. Except for the baud rate, which is changed from the "sane" rate of 9600 to 1200, all I/O parameters are set as if an "stty sane" command had been executed. Sets the timeout value on a read or write call to 20 seconds instead of the 45 seconds (the default value). A carriage return character (hexadecimal 0D) will be appended to each record to indicate end-of-record. No waiting on OPEN is specified. Uses the default value of 1500 characters for the record buffer size.


```
$FTYPE 102,AS $PATH /dev/tty11 $STTY B19200 -ICANON MIN ^a TIME \01
```

Assigns file type identifier 102 to the asynchronous device at path /dev/tty11. Changes the baud rate from 9600 to 19200, disables canonical input processing, and sets the MIN and TIME values to 1. (Note that the stty(1) notation ^a is also supported. This is equivalent to CTRL a, which is a value of 1.) No waiting on OPEN or end-of-record character is specified. Uses the default value of 1500 characters for the record buffer size.

```
$FTYPE 515,AS $PATH /dev/tty11 $WAIT $STTY idevice
```

Assigns file type identifier 515 to the asynchronous device at path /dev/tty11. \$WAIT causes OPENfn to wait for a DTR signal before completing. The "idevice" mode sets I/O parameters to allow reception of data from most alternative keyboard input devices. Uses default values for the record buffer size and timeout, and does not write end-of-record characters.

```
$FTYPE 515,AS $PATH /dev/tty11 $WAIT $STTY -ignbrk brkint ignpar -parmrk  
-inpck istrip -inlcr -igncr icrnl -iucle ixon -ixany ixoff opost  
-olcuc onlcr -ocrnl -onocr -onlret -ofill -ofdel -parenb -parodd cs8  
-cstopb hupcl cread -clocal b9600 isig icanon -xcase -echo -echoe  
-echok -echonl -noflsh
```

Performs exactly the same function as the command in the previous example. In this example, the parameter values for the "idevice" are explicitly set and unset.

```
$FTYPE 515,AS $PATH /dev/tty11 $WAIT $STTY idevice b2400
```

Performs the same function as the command in the previous example, except that the baud rate is changed from the default value of 9600 to 2400.

Section 3
Creating File Type Definitions

```
$FTYPE 715,AS $PATH /dev/tty14 $STTY odevice
```

Assigns file type identifier 715 to the asynchronous device at path /dev/tty14. The "odevice" mode sets I/O parameters to allow data to be written to most 9600 baud printers. Does not write end-of-record characters, and does not wait for DTR on OPEN. Uses default values for record buffer size and timeout.

```
$FTYPE 715,AS $PATH /dev/tty14 $STTY -ignbrk brkint ignpar -parmrk -inpck  
-istrip -inlcr -igncr icrnl -iuclic ixon -ixany ixoff opost -olcuc  
onlcr -ocrnl -onocr -onlret -ofill -ofdel -parenb -parodd cs8 -cstopb  
hupcl cread clocal b9600 -isig -icanon -xcase -echo -echoe -echok  
-echonl -noflsh
```

Performs exactly the same function as the command in the previous example. In this example, the parameter values for the "odevice" are explicitly set and unset.

```
$FTYPE 715,AS $PATH /dev/tty14 $STTY odevice b1200 istrip evenp cstopb
```

Performs the same function as the command in the previous example, except that the baud rate has been changed from 9600 to 1200, and characters are 7 bits, with 2 stop bits. And even parity has been specified.

CREATING A UX (OPERATING SYSTEM) FILE TYPE DEFINITION

The abbreviation UX identifies an operating system file.

To create a UX file type definition using the \$FTYPE supervisory command, enter and execute:

```
$FTYPE nnn,UX $PATH path [$MAXSIZE lll]
```

where

nnn is a three-digit number from 000 to 999 assigned as the identifier to the specified file type definition.

path is the full path name to the operating system file that is to be manipulated. The last component of the path name may be an asterisk (*), indicating that it will be requested when the file is opened. The path name must be less than 80 characters.

lll is the record buffer size, up to 9999 characters. If \$MAXSIZE is not specified, the default value is 1500 characters.

To create a UX file type definition using the FTYPEfn format command, enter:

```
FTYPEfn = "UX $PATH path [$MAXSIZE lll]"
```

where

fn is a number from 1 to 16 assigned as the logical file number to the specified file type definition.

path is the full path name to the operating system file that is to be manipulated. The last component of the path name may be an asterisk (*), indicating that it will be requested when the file is opened. The path name must be less than 80 characters.

lll is the record buffer size, up to 9999 characters. If \$MAXSIZE is not specified, the default value is 1500 characters.

Usage Notes

- The specified file must be either a UNIX text file or a named pipe--special files such as directory files and object code files may not be accessed with MSFP. Additionally, the file may not be in the LiFE-Works database, and it is recommended that you do not create files under the \$W directory.

Section 3

Creating File Type Definitions

- Instead of supplying a literal string on the right side of the FTYPEfn format command, you can generate the right side of the command from another source, such as a field or buffer. For details, see "Generating a File Type Definition into the FTYPEfn Format Command" at the end of this section.
- If you use a literal string rather than a generated value on the right side of the FTYPEfn command, you must enclose the entire right side in single or double quotation marks.
- Even though an operating system (UX) file is simply a stream of bytes, LIFE-Works views it as a file of records. A record consists of all bytes read up to, but not including, a new-line character.

Examples

```
$FTYPE 220,UX $PATH /tmp/exmpl $MAXSIZE 4000
```

Assigns file type identifier 220 to the operating system file at path /tmp/exmpl. Sets the record buffer size to 4000 characters.

```
$FTYPE 004,UX $PATH /tmp/*
```

Assigns file type identifier 004 to the operating system file, whose first path name component is tmp. The last component of the path name will be requested when the file is opened.

```
FTYPE1 = "UX $PATH /usr/text/exmpl"
```

Assigns logical file number 1 to the operating system file at path /usr/text/exmpl.

GENERATING A FILE TYPE DEFINITION INTO THE FTYPEfn FORMAT COMMAND

The following information applies only to the FTYPEfn format command, not to the \$FTYPE supervisory command.

Instead of supplying a literal string on the right side of the FTYPEfn format command, you can generate the file type definition from another source. Use the following syntax:

FTYPEfn = reference

where reference is one of the following:

An	generates from accumulator <u>n</u> .
B(c,l)	generates from the work buffer, starting in column <u>c</u> for a length of <u>l</u> characters.
Fn	generates from field <u>n</u> of the current format.
LFn	generates from local field <u>n</u> of the current subroutine.
Rfn(c,l)	generates from file <u>fn</u> 's record buffer, starting in column <u>c</u> for a length of <u>l</u> characters.
RS(n,l)	generates from a relative screen position, starting <u>n</u> columns from the cursor position at the start of the subroutine for a length of <u>l</u> columns.
S(r,c,l)	generates from the screen area beginning in row <u>r</u> , column <u>c</u> , for a length of <u>l</u> characters.
X(id,c,l)	generates from the selected record in index set <u>id</u> , starting in column <u>c</u> for a length of <u>l</u> characters.
.symbol.	generates from a previously-assigned symbolic name.

Usage Notes

For general information on the FTYPEfn format command, see "About File Type Definitions" at the beginning of this section. For specific command syntax, refer to the information on individual file types presented earlier in this section.

Section 3
Creating File Type Definitions

Examples

```
FTYPE2 = X(100,35,24)
```

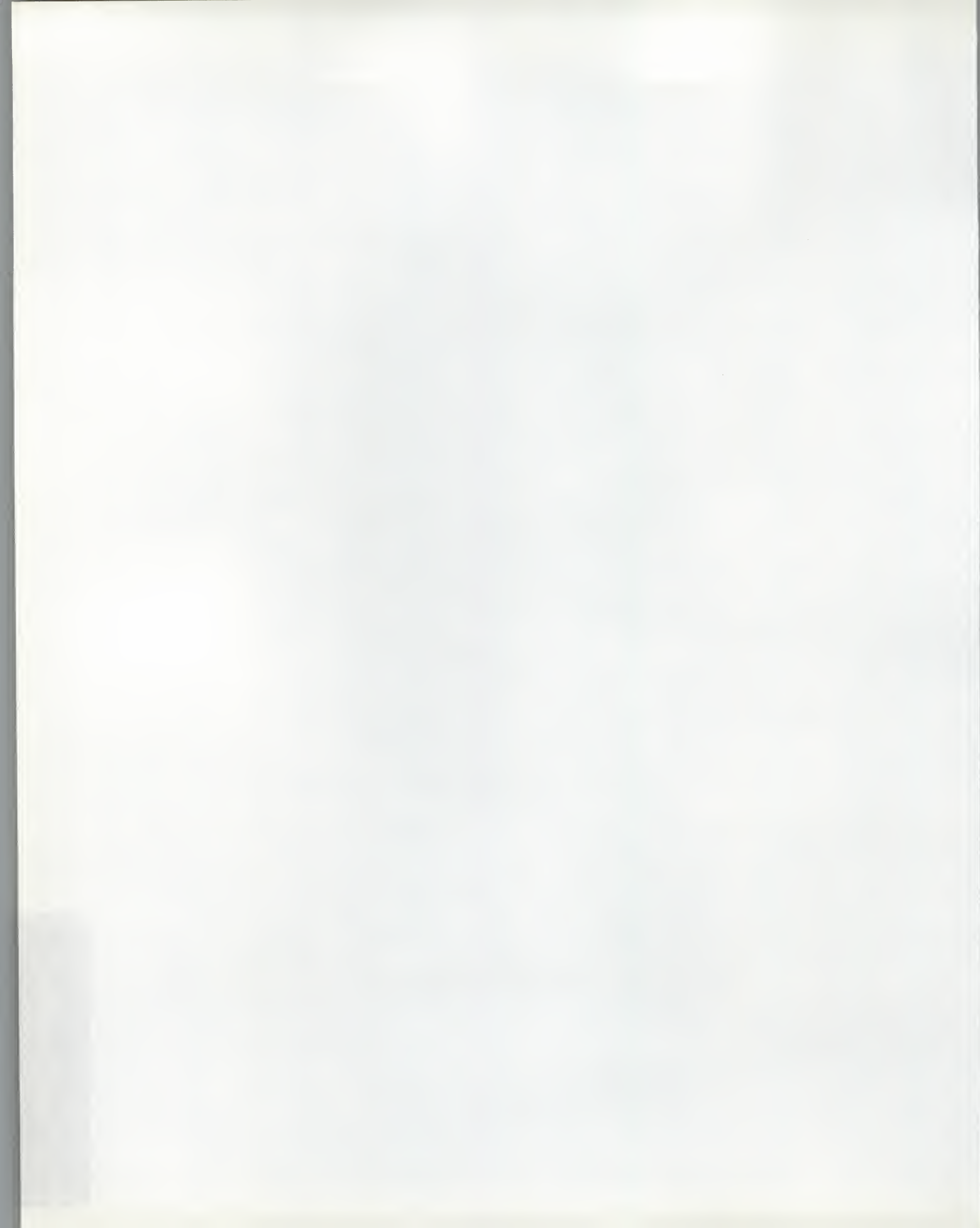
Generates a file type definition from the selected record in index set 100, starting in column 35 for a length of 24 columns; assigns logical file number 2 to that file type definition.

```
P "ENTER FILE TYPE DEFINITION: " 1G47  
FTYPE3 = F1
```

Prompts the operator to enter a file type definition on the screen; assigns logical file number 3 to that file type definition.

```
GETBUF(SIZE=15)  
BLANKB  
SET B(1,10)="XD $INDSET"  
P"ENTER INDEX SET NUMBER:" 2I3  
SET B(12,3)=F2  
FTYPE9=B(1,15)
```

Sets the buffer size to 15 and blanks the buffer. Writes "XD \$INDSET" in the columns 1-10 of the buffer, prompts the operator for an index set number, and writes that number in columns 12-14. Then sets logical file 9 equal to the buffer contents. (This example hides the file type definition from the operator. The operator only needs to know the number of the index set he or she is about to process.)



Section 4
Basic Commands

INTRODUCTION

This section describes the MSFP commands used to manipulate files and records. Table 4-1 lists these commands and explains what each command does. Note that some of the commands apply only to index sets (XD files).

Table 4-1. Basic MSFP Commands

Command	Explanation
CLOSEfn	Closes file <u>fn</u> .
DELETEfn	Deletes the currently selected record from XD file <u>fn</u> . Applies to index sets only.
INSERTfn	Inserts the contents of XD file <u>fn</u> 's record buffer into file <u>fn</u> . Applies to index sets only.
LOCATEfn	Moves the record pointer to a specified position in file <u>fn</u> .
OPENfn	Opens file <u>fn</u> .
READfn	Reads a record into file <u>fn</u> 's record buffer.
RELEASEfn	Releases all access to the currently selected record in XD file <u>fn</u> . Applies to index sets only.
REWRITEfn	Writes the contents of file <u>fn</u> 's record buffer over the record that the record pointer is currently positioned on.
WRITEfn	Writes the contents of file <u>fn</u> 's record buffer into file <u>fn</u> .

Usage Notes on Index Sets

- Any of the commands that apply only to index sets abort the format if file fn is not an XD file.
- If you open an XD file for input, you get shared access to each record. If you open an XD file for output, you get exclusive access to each record.

Section 4
Basic Commands

- MSFP index set commands do not automatically relinquish access when the operator releases the screen record. To give other terminals access to the index set record, you must use RELEASEfn or the R option of the WRITEfn or REWRITEfn commands. It is good practice to release an index set record as soon as you no longer need it.
- The RECSELfn conditional indicator lets you test whether a record is currently selected from index set fn. RECSELfn is discussed in Section 6.
- In many cases you can choose between MSFP and non-MSFP methods for working with index sets.
- Opening an XD file in VISION on the Series 5000 always positions the record pointer at BOF. This is inconsistent with other file types. In LIFE-Works on the Series 8000, the opening of a file is the same regardless of the file type. The rules are as follows:
 - Opening for input always positions the record pointer at BOF.
 - Opening for output always positions the record pointer at EOF if the file contains records. If the file is empty, the record pointer is positioned at BOF.

XD files are not treated as a special case. If the format code logic requires an index set with records to be positioned at BOF after an OPENfn, then use you use the "R" (rewind) option with OPENfn, or execute LOCATEfn BOF after you open the file.

THE CLOSEfn COMMAND

Use the CLOSEfn command to close file fn when processing is complete. The CLOSEfn command syntax is as follows.

For SD files:

```
CLOSEfn,[C][,L]
```

For XD, AS, and UX files:

```
CLOSEfn
```

For PR files:

```
CLOSEfn[,D]
```

where

fn is the logical file number of the file being closed.

C sets the COMP flag on the closed file. Applies only to SD files. Ignored for other file types.

L sets the LOCK flag on the closed file. Applies only to SD files. Ignored for other file types.

D delays printing--that is, it closes the PR file but holds the spool file out of the print queue. Applies only to PR file types opened with the permanent spool file [S] option. Ignored for other file types and for PR files opened with the temporary spool file [T] option.

Usage Notes

- The CLOSEfn command is ignored if file fn is not open.
- When CLOSEfn is executed:
 - The contents of file fn's record buffer are no longer accessible to the format.
 - The conditional indicators OPENEDfn, OPENINfn, and OPENOUTfn are set to false.

Section 4
Basic Commands

- If file fn is a PR file, a page eject is placed in the spool file before the PR file is closed.
- Use the Current Spool File Display (mode P-S) to print a permanent spool file that has been held out of the print queue. See Section 8 for information on the Current Spool File Display.

Examples

```
CLOSE1,C,L
```

Closes logical file 1. If the file is an SD file, sets the COMP and LOCK flags.

```
IF #BATCH6="FRIDAY" THEN CLOSE6 ELSE NULL
```

Closes logical file 6 if the file is named FRIDAY; otherwise, takes no action.

THE DELETefn COMMAND

Use the DELETefn command to delete the currently selected record from XD file fn. The DELETefn command syntax is as follows.

For XD files:

DELETefn

For SD, UX, PR, and AS files:

Not applicable; aborts the format.

where

fn is the logical file number of the file containing the record to be deleted.

Usage Notes

- To delete an index set record, you must have exclusive access (that is, the XD file must be open for output), and the record must be selected (RECSEL must be true). The record can be selected by INSERTfn or by any form of the READfn command. Note that any READfn command reads the record into file fn's record buffer, and that DELETefn does not affect the contents of the record buffer.
- DELETefn aborts the format if file fn is not an XD file, file fn is opened for input only, or RECSELfn is false.
- Executing DELETefn sets RECSELfn false.
- Deleting the first record of the file sets BOFfn true.
- In Release VA01 of VISION II and in VISION on the Series 4000/5000, if you attempt to delete a record that others are waiting for, the format aborts with an error message. Effective in Release VA02 of VISION II and in Release VA03 of LIFE-Works, this situation no longer exists. You can delete a record that others are waiting for. If you do, those waiting for the record get "no hit" or "no record selected" and the format continues.

Section 4
Basic Commands

Examples

```
IF OPENOUT3&RECSEL3 THEN DELETE3 ELSE NULL
```

If you have exclusive access to logical file 3 (OPENOUTfn is true) and a record is selected (RECSELfn is true), deletes the selected record from logical file 3.

```
READ1(KEY=F1)  
IF RECSEL1 THEN DELETE1 ELSE SET S(22,25,16)="RECORD NOT FOUND"
```

Selects and reads the record from logical file 1 whose key-field value is in field 1, then deletes the selected record from logical file 1. If the record does not exist (is not selected), the appropriate message is set on the screen.

THE INSERTfn COMMAND

Use the INSERTfn command to insert the record contained in XD file fn's record buffer into the file, based on the key in the record. The INSERTfn command syntax is as follows.

For XD files:

INSERTfn[(key)]

For SD, UX, PR, and AS files:

Not applicable; aborts the format.

where

fn is the logical file number of the file into which the record is to be inserted.

key is the name of the key field to be used for record selection. If key is omitted, the primary key is used.

Usage Notes

- INSERTfn aborts the format if XD file fn is open for input only.
- INSERTfn[(key)] selects the inserted record based on the specified key. If no key is specified, the selection is based on the primary key. After INSERTfn executes, the inserted record is still exclusively accessed and RECSELfn is set true.
- The program level and record length of the inserted record are determined by the current values of #PLEVELfn and #RECLFNfn.
- The MSFP INSERTfn command is different from the INSIXR format programming command. INSIXR inserts a blank record into an index set. In contrast, when INSERTfn is executed, whatever is in the record buffer is inserted into the index set.
- If the INSERTfn(key) command uses a key field that doesn't allow duplicates and if a record with that key already exists in the file, INSERTfn(key) aborts the format. See the discussion of \$INDSET command in the LIFE-Works Format Language Manual, Volume 1 for information on how to allow for duplicate keys.

Section 4
Basic Commands

Examples

```
INSERT4
```

Inserts the record in logical file 4's record buffer into logical file 4.

```
READ4(KEY=F1)  
IF !RECSEL4 THEN INSERT4 ELSE NULL
```

Reads from logical file 4 a record whose key equals field 1. If the record is not found (is not selected), inserts the record in file 4's record buffer into file 4.

```
IF R2(1,5)!X(500,1,5) THEN INSERT2 ELSE NULL
```

If the first five characters of logical file 2's record buffer are not equal to the first five characters of the currently selected record in index set 500 then inserts the contents of record buffer 2 into logical file 2.

```
READ3(KEY(SSN)=F1)  
IF !RECSEL3 THEN INSERT3(SSN)
```

Reads from logical file 3 a record whose key, SSN (for social security number), has a value equal to field 1. If the record is not found (is not selected), inserts the record in file 3's record buffer into file 3. The inserted record is selected based on the key SSN.

THE LOCATEfn COMMAND

Use the LOCATEfn command to move the record pointer to a specific position in file fn. The LOCATEfn command syntax is as follows.

For SD, XD, and UX files:

LOCATEfn	$\left\{ \begin{array}{l} \text{BOF} \\ \text{EOF} \\ \text{expression} \end{array} \right\}$
----------	---

For PR and AS files:

Not applicable; aborts the format.

where

fn is the logical file number of the file.

BOF moves the record pointer to beginning-of-file, before the first record.

EOF moves the record pointer to end-of-file, after the last record.

expression is evaluated at run time. A positive value moves the record pointer forward by the number of records specified; a negative value moves the record pointer backward by the number of records specified; expression can be an arithmetic expression, an absolute value, or a value generated from any of these sources:

An	generates a value from accumulator <u>n</u> .
B(c,l)	generates a value from the work buffer, starting in column <u>c</u> for a length of <u>l</u> characters.
Fn	generates a value from field <u>n</u> of the current format.
LFn	generates a value from local field <u>n</u> of the current subroutine.
Rfn(c,l)	generates a value from file <u>fn</u> 's record buffer, starting in column <u>c</u> for a length of <u>l</u> characters.
RS(n,l)	generates a value from a relative screen position, starting <u>n</u> columns from the cursor position at the start of the subroutine for a length of <u>l</u> columns.
S(r,c,l)	generates a value from the screen area beginning in row <u>r</u> , column <u>c</u> , for a length of <u>l</u> characters.

Section 4

Basic Commands

`X(id,c,l)` generates a value from index set id, starting in column c for a length of l characters.

`.symbol.` generates from a previously-assigned symbolic name.

Usage Notes

- You can't use LOCATEfn in a PR or AS file. If you try, the format aborts.
- LOCATEfn positions the record pointer; it does not read or write records. LOCATEfn does not affect the #RECNUMfn, #RECLFNfn, and #PLEVELfn descriptors.
- READfn reads the record following the located record, and WRITEfn writes over the record following the located record. For example, if LOCATEfn positions the record pointer on record 5 of file fn, the next READfn reads record 6, or the next WRITEfn writes over record 6.
- REWRITEfn writes over the located record. For example, if LOCATEfn positions the record pointer on record 5 of file fn, the next REWRITEfn writes over record 5.
- If the program finds the beginning or end of the file before locating the record specified in expression, the following things happen:
 - LOCATEfn terminates.
 - The appropriate conditional indicators (BOFfn, BOBfn, EOFfn, EOBfn) are set true.
 - The next format statement executes.
- In an SD file that is an order chain, locating forward and locating to end-of-file and beginning-of-file work the same way as in any other file. Locating backward doesn't work the same way as in other files. You cannot locate backward past the beginning of the current batch in an order chain.

For example, if there are three batches in an order chain, and the record pointer is on record 3 of the second batch, executing LOCATE -6 doesn't move the record pointer into the first batch. Instead, the record pointer stops at the beginning of the second batch.

You can determine when you have hit a batch boundary in an order chain by checking the BOBfn and EOBfn conditional indicators. When you locate forward in an order chain, EOBfn is set true every time you cross a batch boundary. When you locate backward in an order chain, BOBfn is set true when you encounter (and stop at) a batch boundary. BOBfn and EOBfn are described in Section 6.

- In an XD file, RECSELfn is always set false after a locate.

- You can't use an absolute record number as a LOCATEfn parameter, but you can use #RECNUMfn with LOCATEfn to move to a specified record. For example, to locate to record number 17 of file 5, you would use the command

LOCATE5 17-#RECNUM5

Examples

LOCATE1 BOF

Positions the record pointer before the first record in logical file 1.

LOCATE4 +5

Positions the record pointer five records forward in logical file 4.

LOCATE5 A1

Positions the record pointer in logical file 5 according to the value contained in accumulator 1 at run time.

Section 4
Basic Commands

THE OPENfn COMMAND

Use the OPENfn command to open a file. The syntax of the OPENfn command depends on the type of file being opened and whether the file was defined with the \$FTYPE supervisory command or the FTYPEfn format command. The command syntax is as follows.

For SD files:

$$\text{OPENfn}, \left\{ \begin{matrix} \text{nnn} \\ e \end{matrix} \right\}, \left\{ \begin{matrix} I \\ O \end{matrix} \right\} [,C][,R]$$

For XD files:

$$\text{OPENfn}, \left\{ \begin{matrix} \text{nnn} \\ e \end{matrix} \right\}, \left\{ \begin{matrix} I \\ O \end{matrix} \right\} [,R]$$

For PR files:

$$\text{OPENfn}, \left\{ \begin{matrix} \text{nnn} \\ e \end{matrix} \right\}, O[,NC=ccc] \left[\begin{matrix} ,S["spoolfile"] \\ ,T["spoolfile"] \end{matrix} \right]$$

For AS files:

$$\text{OPENfn}, \left\{ \begin{matrix} \text{nnn} \\ e \end{matrix} \right\}, \left\{ \begin{matrix} I \\ O \end{matrix} \right\} [,TO=sec]$$

For UX files:

$$\text{OPENfn}, \left\{ \begin{matrix} \text{nnn} \\ e \end{matrix} \right\}, \left\{ \begin{matrix} I \\ O \end{matrix} \right\} [,C][,R][,P=ppp]$$

where

fn is a logical file number (1-16).

nnn is the three-digit file type identifier used in the \$FTYPE supervisory command. Use nnn only if the file type was previously defined with the \$FTYPE command.

@ indicates that the file type definition for file fn appears earlier in the format program. Use @ only if the file type is defined in this format with the FTYPE format command.

I opens the file for input. A file opened for input can be read from but not written to. You must specify either I or O in each OPENfn command. The I or O must be the first parameter after OPENfn, as shown above.

O opens the file for output. A file opened for output can be read from and written to. You must specify either I or O in each OPENfn command. The I or O must be the first parameter after OPENfn, as shown above.

C clears all records from the file before it is opened for output. Applies only to SD and UX files and is ignored for all other file types.

R "rewinds" the file--that is, positions the record pointer before the first record in the file. Applies to disk files opened for output; ignored for AS and PR files and for any file opened for input. R sets BOFfn (and BOBfn if it applies) true.

ccc indicates the number of copies to be printed, where ccc is a number from 1 to 999. Applies only to PR files and is ignored for all other file types. If you omit NC, one copy is printed.

"spoolfile" spools printer output to a permanent spool file (S) or a temporary spool file (T). spoolfile is the spool file name (1-14 characters, enclosed in quotation marks). Applies only to PR files and is ignored for other file types. If you include S or T but omit the spool file name, the system generates a name. If you don't include either S or T, MSFP uses a temporary spool file with a system-generated name. (See Section 8 for detailed information on spool files.)

sec is the number of seconds, up to a maximum of 999999, that can pass before the OPENfn command is interrupted. A value of 0 means "do not timeout." Applies to AS files only. If you omit the TO sec option when you open an AS file, the timeout value as specified in the ftype definition for that file is used. If you specified a \$TIMEOUT value in the ftype definition for the file, the TO sec option in the OPENfn command overrides that value.

ppp is the octal representation of the permissions that are to be used if the file is created. If ppp is omitted and the file is created, the default file permissions are set to 644, which indicates read and write permissions for the owner and read-only permissions for everyone else. (For additional information on permissions, see chmod(1) in the System V/68 User's Reference Manual.)

Generating OPENfn Command Parameters

You can generate many of the OPENfn command parameters from another source. Use the following syntax:

$\text{OPENfn,ref1,} \left\{ \begin{array}{c} \text{I} \\ \text{O} \end{array} \right\} [\text{,C}][\text{,NC=ref2}][\text{,R}] \left[\begin{array}{c} \text{,S=ref3} \\ \text{,T=ref3} \end{array} \right] [\text{,TO=ref4}][\text{,P=ref5}]$

where

ref1 generates a three-digit file type identifier (000-999). The file type definition must exist and must have been created using \$FTYPE. ref1 can be any of these sources:

- An generates a value from accumulator n.
- B(c,l) generates a value from the work area buffer, starting in column c for a length of l characters.
- Fn generates a value from field n of the current format.
- LFn generates a value from local field n of the current subroutine.
- Rfn(c,l) generates a value from file fn's record buffer, starting in column c for a length of l characters.
- RS(n,l) generates a value from a relative screen position, starting n columns from the cursor position at the start of the subroutine for a length of l columns.
- S(r,c,l) generates a value from the screen area beginning in row r, column c, for a length of l characters.
- X(id,c,l) generates a value from index set id, starting in column c for a length of l characters.
- .symbol. generates from a previously-assigned symbolic name.

ref2 generates the number of copies to be printed. ref2 can be any of the sources described for ref1. The generated value can be up to three digits long (0-999). If the generated value is zero, one copy is printed.

ref3 generates a permanent [S] or temporary [T] spool file name. The generated value can be up to 14 characters long. ref3 can be any of these sources described for ref1.

ref4 generates a timeout value, from 1 to 6 digits. ref4 can be any of the sources described for ref1.

ref5 generates an octal value representing the file permissions. ref5 can be any of the sources described for ref1.

The remaining OPENfn command parameters are the same for all forms of the command. See descriptions earlier in this section.

Usage Notes

- Before you can do any processing in an MSFP file, you must use OPENfn to open the file. The file stays open until:
 - You execute a CLOSEfn for the file.
 - The operator presses EXIT.
 - The operator presses RESET after a runtime error.
- When you open a file that has been defined with the \$FTYPE supervisory command, the OPENfn command must identify the file type definition (by its three-digit identifier) and assign a logical file number to it. You then use the logical file number to refer to that file type definition throughout the format code.

When you open a file that has been defined with the FTYPEfn format command, the logical file number has been assigned in the FTYPEfn command. You use the assigned logical file number in the OPENfn command and throughout the rest of the code.

- MSFP allocates a record buffer for each file opened. Each record buffer's length is equal to the maximum record size defined for the corresponding file. (See Section 5 for details.) The record buffer is initialized to blanks.
- Opening a file for input sets the following values:
 - OPENEDfn is set true.
 - OPENINfn is set true.
 - OPENOUTfn is set false.
 - BOFfn is set true for all files except AS files. It is set false for AS files.
 - BOBfn is set false for all files except SD files. It is set true for SD files.
 - EOFfn is set false.
 - EOBfn is set false.

Section 4
Basic Commands

- #PLEVEL is set to 1.
- #RECNUM is set to 0.
- Opening a file for output sets the following values:
 - OPENEDfn is set true.
 - OPENINfn is set false.
 - OPENOUTfn is set true.
 - BOFfn is set true if the file is empty (new or opened with the clear [C] option) or if the file is opened with the rewind [R] option. BOFfn is set false if the file contains records and is not opened with the rewind option. BOFfn is also set false for AS files.
 - EOFfn is set true if BOFfn is false and vice versa.
 - EOFfn is always set false for AS files.
 - #PLEVEL is set to 1.
 - If EOFfn is true, #RECNUM is set to the total number of records in the file for all files except XD and AS files. It is set to zero for XD and AS files.
 - If BOFfn is true, #RECNUM is set to zero.
- Notes on opening SD files:
 - When you open an SD file for input, the specified batch must already exist; if it doesn't, the format aborts. When you open an SD file for output, you can specify either an existing batch or a nonexistent batch. If the batch is nonexistent, it will be created automatically.
 - Any number of MSFP users can have the same batch open for input, but only one user can have a given batch open for output. When MSFP has a batch open, only other MSFP users can access the batch; the batch cannot be accessed through entry, find, or verify modes. If the job is password-protected, the password is requested when the file is opened.
 - You cannot open an SD file if any of the following flags are set on the batch: RRF, RWR, RRD, INPR, RPUR, or XSET.
 - Opening an SD file sets the following values:
 - BOBfn is set to the same value as BOFfn.
 - EOBfn is set to the same value as EOFfn.

- #JOBfn is set to the job name specified in \$FTYPE or FTYPEfn.
 - #BATCHfn is set to the batch name specified in \$FTYPE or FTYPEfn. If the batch name is entered at run time, #BATCHfn is set to the entered name. If the SD file is an order chain, #BATCH is set to the name of the first or last batch in the chain, depending on how the record pointer is positioned.
 - #FILEfn is set to blanks.
- Notes on opening XD files:
 - The index set must exist and must have been defined with the \$INDSET supervisory command. If you try to open a nonexistent index set, the format aborts.
 - Any number of users can have the same index set open, using either MSFP or non-MSFP methods. A single MSFP program can have the same index set open more than once, but be aware of record contention problems. For example, logical file 1 could have exclusive access to a record while logical file 2 is waiting for that record, so the program ends up waiting on itself.
 - When you open an XD file for input, you obtain shared access to each record read. When you open an XD file for output, you obtain exclusive access to each record read.
 - Opening an XD file sets the following values:
 - #JOBfn is set to the job specified in the \$IXBATCH parameter of the \$INDSET command.
 - #BATCHfn is set to the batch specified in the \$IXBATCH parameter of the \$INDSET command.
 - #FILEfn is set to blanks.
 - #RECLenfn is initialized to the maximum record length of the job specified in the \$INDSET command.
 - #RECNUMfn is initialized to 0.
 - Notes on opening PR files:
 - PR files can be opened only for output. The format aborts if you try to open a PR file for input.
 - Opening a PR file sets the following values:
 - #JOBfn is set to blanks.

Section 4
Basic Commands

- #BATCHfn is set to the first six characters of the spool file name.
- #RECLFN is set to the length of the print line.
- #FILEfn is set to the spool file name.
- See Section 8 for additional information on spool files and other topics related to printing.
- Notes on opening AS files:
 - Opening an AS file sets the following values:
 - #JOBfn is set to blanks.
 - #BATCHfn is set to blanks.
 - #RECLFN is initialized to 80.
 - #RECNMfn is initialized to 0.
 - #FILEfn is set to contain the last component of the path name to the device.
 - A value given in the TO sec option applies only to the OPENfn command. READfn and WRITEfn commands still adhere to the timeout value specified by the \$TIMEOUT parameter in the ftype definition.
- Notes on opening UX files:
 - If the OPENfn command causes the UX file to be created, the file's owner id and group id are both set to "vision."
 - When you open a UX file for input, the specified file must already exist; if it doesn't, the format aborts. When you open a UX file for output, you can specify either an existing file or a nonexistent file.
 - Any number of MSFP users can have the same file open for input, but only one user can have a given file open for output. When MSFP has a file open, only other MSFP users can access that file.
 - Opening a UX file sets the following values:
 - #JOBfn is set to blanks.
 - #BATCHfn is set to blanks.
 - #RECLFN is initialized to 80.

- #RECNUM is set to zero if the file is opened at the beginning-of-file. Otherwise, it is set to the total number of records in the file.
- #FILEfn is set to the last component of file's path name.

Examples

```
OPEN1,223,0,C
```

Assigns logical file number 1 to Ftype 223 and opens the file for output. If Ftype 223 is an SD file, clears all existing data from the file.

```
OPEN6,300,0,R
```

Assigns logical file number 6 to Ftype 300, opens the file for output, and positions the record pointer to beginning-of-file.

```
OPEN2,@,I
```

Opens logical file 2 for input. An FTYPEfn command earlier in the format identified the file type definition and assigned logical file number 2 to it.

```
OPEN3,S(2,1,3),0,NC=S(4,1,3),S=S(6,1,14)
```

Generates a file type identifier from the screen, assigns logical file number 3 to that file type definition, and opens the file for output. Generates the number of copies to be printed and the name of a permanent spool file from the screen.

Section 4
Basic Commands

THE READfn COMMAND

Use the READfn command to read a record from file fn into the record buffer associated with file fn. The READfn command syntax is as follows.

For SD and UX files:

```
READfn[,R]
```

For XD files:

```
READfn[,R]
```

```
READfn(KEY[(key)]=reference)[,R]
```

```
READfn(KEY[(key)]>=reference)[,R]
```

For AS files:

```
READfn [ (RECLen=n) [ ,NW ]  
        (UNTIL=c) ]
```

For PR files:

Not applicable; aborts the format.

where

fn is the logical file number of the file being read.

R "rewinds" the file before reading--that is, moves the record pointer to beginning-of-file, before the first record. Not applicable to AS files.

key is the name of the key field to be used. If key is omitted, the primary key is used. Applies to XD files only.

reference generates the value you want the key field to be equal to or greater than. The generated value must be the same length as the key field. reference can be any of these:

An generates a value from accumulator n. Note that the key field must equal the accumulator length (12 or 24 characters).

B(c,l) generates a value from the work buffer, starting in column c for a length of l characters.

F_n generates a value from field n of the current format.

LF generates a value from local field n of the current subroutine.

Rfn(c,l) generates a value from file fn's record buffer, starting in column c for a length of l characters.

RS(n,l) generates a value from a relative screen position, beginning n columns from the cursor position at the start of the subroutine and continuing for l columns.

S(r,c,l) generates a value from the screen area beginning in row r, column c, for a length of l characters.

X(id,c,l) generates a value from index set id, beginning in column c for a length of l characters.

.symbol. Generates from a previously-assigned symbolic name.

(RECLEN=n) reads n characters into the record buffer. Note that n can be generated from any of the system elements defined for reference above. Applies only to AS files.

(UNTIL=c) reads characters into the record buffer until the character specified by c is received. Use this option to detect a \$EORM end-of-record character. (See "Creating an AS File Type Definition" in Section 3.) Note that c can be generated from any of the system elements defined for reference above. Applies only to AS files. Typically, c is specified as the hexadecimal code for the character. However, if it is a displayable character, you may specify the character itself enclosed in quotes. (See "Examples" below.)

NW indicates that the READ_{fn} command is not to wait for data to arrive. If READ_{fn},NW does complete before receiving data, it sets #RECL_{EN} to zero. Note that if NW is not specified, READ_{fn} waits for data before completing. This causes the command to become "blocked" until data is received. Applies only to AS file types.

Usage Notes

- You cannot read from a printer file. If file fn is a PR file, READ_{fn} aborts the format.
- READ_{fn}(KEY[(key)]=...) and READ_{fn}(KEY[(key)]>=...) can be used only with XD files. The format aborts if these commands are used with other file types.

- When READfn executes:
 - The record following the record pointer is read into the record buffer.
 - The record pointer is moved forward one record.

For example, if the record pointer is positioned at record 10 of file fn, READfn reads record 11 into the record buffer, then moves the record pointer to record 11.

As you code your format, keep in mind that READfn reads the next record, not the record at which the record pointer is currently positioned.

- After reading a record into the record buffer, you can manipulate the data in the buffer by using SET statements and the Rfn descriptor. (Rfn is described in Section 5.) The data is accessible until a CLOSEfn is executed on the file, or until the operator presses EXIT or QUIT. The next READfn executed on the file overwrites the data from the previous READfn.
- READfn sets the following values:
 - #PLEVELfn is set to the program level of the record read for all files except AS and UX files. #PLEVELfn is always set to 1 for AS and UX files, even though it is never referenced.
 - #RECLEfn is set to the length of the record just read.
 - #RECNUMfn is set to the record number of the record read for SD and UX files. For AS files, it is initialized to zero and incremented by one for each READfn. (For XD files, the value of #RECNUMfn depends on the type of READ done. See "The #RECNUM System Keyword" in Section 7 for details.)
- Notes on reading XD files:
 - Three forms of the READfn command can be used with index sets. The READfn command (without a key) lets you read index set records sequentially, an operation that can be done only through MSFP. The two "READfn with key" options let you read an index set record by specifying a key value.
 - Any form of the index set READfn command gives the initiating terminal access to the record until the next READfn, WRITEfn,R, REWRITEfn,R, LOCATEfn, CLOSEfn, or RELEASEfn executed in the same file, or until the operator exits the current batch. If the file is open for output, the terminal has exclusive access. If the file is open for input, the terminal has shared access.
 - If another terminal has exclusive access to the record specified in the READ command, a message displays and the program waits. When the record is released, the program selects and reads the record. (If the operator prefers not to wait, he or she can press QUIT to abort the program.)

- If READfn is executed when BOFFn is true, the first record is read. If READfn is executed after the last record is read, the record buffer remains unchanged, RECSEfn is set false, and EOFfn is set true.
- If the READfn(KEY[(key)]=...) command doesn't find a record whose key equals the specified value, the following things happen:
 - No record is read, and the record buffer remains unchanged.
 - #PLEVfn and #RECLfn remain unchanged.
 - RECSEfn is set false.
- If the READfn(KEY[(key)]>=...) command doesn't find a record whose key equals the specified value, it selects and reads the record with the next higher key value. If the specified value is higher than any key value in the index set, READfn(KEY[(key)]>=...) selects and reads the record with the highest key value.
- You must use either the RELEASEfn command or the R subcommand of the WRITEfn or REWRITEfn command to release access to an index set record.
- Notes on reading AS files:
 - If neither (RECLN=n) nor (UNTIL=c) is specified for an AS file, READfn reads all characters until the system buffer is flushed. The flush occurs when one of the following happens:
 - the buffer is full
 - a new-line (carriage return) is received
 - if -icanon has been specified as an \$STTY mode, and "MIN" characters have been read or "TIME" has expired. (See termio(7) in the System V/68 System Administrator's Reference Manual for additional details.)
 - There is no way to distinguish between the operator aborting the format (pressing CTRL RESET) and a break condition on the line. The operating system translates the break to an interrupt signal, which is same signal generated by pressing CTRL RESET.
 - In instances where system performance an issue, it is slightly more efficient to allow the READfn to operate in its default state, rather than overriding it with the NW option.
 - If the right side of the UNTIL parameter is a reference that contains multiple characters, only the first character of the field is considered the "until character." The rest of the field is ignored.

Section 4
Basic Commands

- When UNTIL=c is used with the READfn command, the c character is placed in the MSFP record buffer, but it is not included as part of the record length count.
- Notes on reading UX files:
 - Since UX files contain variable length records, READfn sets #RECLen to the number of characters in the record just read.
 - A read operation fills the record buffer with all characters up to, but not including, the new-line character which marks the end of the record. #RECLen is set to the number of characters put in the record buffer.

Examples

```
READ4
```

Reads a record from logical file 4.

```
IF OPENED3 THEN READ3 ELSE F1
```

If logical file 3 is open, reads a record from the file. If logical file 3 is not open, goes to program level 1.

```
READ4 (RECLen=96)
```

Reads a 96-character record from AS logical file 4.

```
READ2 (UNTIL=\13)
```

Reads characters from AS logical file 2 until an XOFF (DC3) character is received. (The hexadecimal representation of an XOFF character is 13.)

READ3 (UNTIL="X")

Reads characters from AS logical file 3 until the character X is received.

READ5,NW

Reads characters from AS logical file 5. Does not wait for data.

READ7(KEY=F10)

Reads the record in logical file 7 whose key field is equal to the contents of field 10.

READ5(KEY=R2(15,8))

Reads the record in logical file 5 whose key field is equal to the contents of columns 15-22 in logical file 2's record buffer.

READ4(KEY(ZIP)>=S(2,5,5))

Reads the record in logical file 4 whose key field, ZIP, is equal to or greater than the value in row 2, columns 5-9 of the screen.

READ3
READ12(UNTIL=R3(56,1))

Reads a record from logical file 3. Reads data from the port (AS file) defined as logical file 12 until the character in position 56 of logical file 3's record buffer is encountered.

THE RELEASEfn COMMAND

Use the RELEASEfn command to release the initiating terminal's access to a selected index set record in file fn. The RELEASEfn command syntax is as follows.

For XD files:

```
RELEASEfn
```

For SD, UX, PR, and AS files:

Not applicable; aborts the format.

where

fn is the logical file number of the index set being accessed.

Usage Notes

- You must use either the RELEASEfn command or the R subcommand of the WRITEfn or REWRITEfn command to release access to an index set record.

Examples

```
RELEASE4
```

Releases all access to the currently selected record in logical file 4.

```
READ1(KEY=F1)  
IF !RECSEL1 THEN (SET R1(1,5)=F1 INSERT1 RELEASE1) ELSE RELEASE1
```

Reads the record from logical file 1 whose key field matches the value in field 1, then releases file 1. If a key field match cannot be found, inserts a new record into file 1, then releases the record.

THE REWRITEfn COMMAND

Use the REWRITEfn command to rewrite the last record read from file fn. The REWRITE command syntax is as follows.

For SD, UX, PR, and AS files:

REWRITEfn

For XD files:

REWRITEfn[,R]

where

fn is the logical file number of the file being written to.

R releases all access to the index set after the write. Applies only to XD files. Ignored for other file types.

Usage Notes

- For AS, PR, and XD files, REWRITEfn performs the same function as WRITEfn.
- For SD and UX files, REWRITEfn is similar to WRITEfn, but REWRITEfn overwrites the current record (the record on which the record pointer is positioned) rather than overwriting the next record. REWRITEfn simplifies some writes to SD and UX files. For example, to overwrite a record using WRITEfn, you would use a command sequence like this:

```
LOCATE1 BOF /* Positions the record pointer at beginning-of-file */
READ1      /* Reads record 1 and moves the record pointer to record 1 */
LOCATE1 -1 /* Moves the record pointer back to beginning-of-file */
WRITE1     /* Writes over record 1 */
```

If you use REWRITEfn, you don't have to back up:

```
LOCATE1 BOF /* Positions the record pointer at beginning-of-file */
READ1      /* Reads record 1 and moves the record pointer to record 1 */
REWRITE1   /* Writes over record 1 */
```

Section 4
Basic Commands

- REWRITEfn fails if the record pointer is at beginning-of-file or end-of-file.

Examples

```
REWRITE1
```

Writes the contents of file 1's record buffer over the record on which the record pointer is currently positioned.

```
REWRITE6,R
```

Writes the contents of XD file 6's record buffer over the record on which the record pointer is currently positioned. Releases access to the index set.

```
READ13  
IF !EOF  
THEN (IF R13(20,2)="04"  
      THEN (SET R13(20,2)="99" REWRITE13)  
      ELSE F6)  
ELSE F8
```

Reads a record from logical file 13. If this is not the end-of-file, checks the condition code in the record. If the condition code is 4, changes the code to 99 and rewrites the record. Otherwise, goes to program level 6 for processing of the record. If this is the end-of-file, goes to program level 8.

THE WRITEfn COMMAND

Use the WRITEfn command to write the contents of file fn's record buffer into file fn. The WRITEfn command syntax is as follows.

For SD and UX files:

```
WRITEfn
```

For XD files:

```
WRITEfn[,R]
```

For PR files:

```
WRITEfn[,Cc][,CR][,Ss]
```

For AS files:

```
WRITEfn[(RECLen=n)][,BR]
```

where

fn is the logical file number of the file being written to.

R releases all access to the index set record after the write. Applies only to XD files; ignored for other file types.

Cc causes a printer to skip to carriage control channel c (1-12) before printing the contents of the buffer. Applies only to PR files; ignored for other file types. If channel c is not defined, the WRITEfn command aborts. (For information on carriage control channels, see Section 8.)

CR suppresses the line feed on a printer after the buffer has been printed. This means that the next line written to the printer will print over this line. Applies only to PR files; ignored for other file types.

Ss causes a printer to skip s lines (1-15) before printing the contents of the buffer. Applies only to PR file types; ignored for other file types. If skipping s lines would move the form beyond the last print line on a page, a top-of-form is done instead of the specified skip. To skip more than 15 lines, use multiple WRITEfn commands.

(RECLFN=n) writes n characters from the record buffer. Note that n can be generated from any of the system elements defined for reference under the READFN command. Applies only to AS files.

BR causes a break condition to be sent. Applies only to AS files; ignored for other file types.

Usage Notes

- WRITEFN writes the entire record buffer. If the file contains variable-length records, the amount of data written is up to the value of RECLFN or #RECLFN.
- For SD files with variable-length records and for UX and XD files, the current value of #RECLFN determines the output record length.
- For SD and XD files, the current value of #PLEVELFN determines the output program level.
- WRITEFN does not change the contents of the record buffer.
- Notes on writing to SD files:
 - WRITEFN can add a record to the end of a batch or overwrite an existing record. It cannot insert a record into a batch or delete a record from a batch.
 - If the record pointer is on the last record or at end-of-file, WRITEFN adds the record to the end of the batch.
 - If the record pointer is on any record except the last record, WRITEFN overwrites the record following the record pointer (not the current record).
 - The REWRITEFN command simplifies some writes to SD files. REWRITEFN is described in this section.
- Notes on writing to XD files:
 - WRITEFN writes over the record to which you currently have exclusive access. (In other words, WRITEFN performs a rewrite.) If no record is exclusively accessed, the command aborts.
 - You cannot change the value of the primary key field by using the WRITEFN command. WRITEFN aborts if the key field and the corresponding part of the record buffer don't match.
 - To add records to an index set, see the information on the INSERTFN command in this section.

- Notes on writing to PR files:

- The Cc, Ss, and CR options, in combination with Vertical Format Unit (VFU) simulation, give you precise control over a printer. See Section 8 for information on VFU simulation.
- You cannot specify both Cc and Ss in the same WRITEfn statement.

- Notes on writing to AS files:

- When the BR option is specified, WRITEfn simply raises a break condition on the line. The contents of the record buffer are not written.
- There is no way to distinguish between the operator aborting the format (pressing CTRL RESET) and a break condition on the line. The operating system translates the break to an interrupt signal, which is same signal generated by pressing CTRL RESET.
- If you specify (RECLen=n) in a WRITEfn command, it takes precedence over the value of #RECLenfn.

- Notes on writing to UX files:

- WRITEfn can add a record to the end of a file or overwrite an existing file. It cannot insert a record into a file or delete a record from a file.
- If the record pointer is on the last record or at end-of-file, WRITEfn adds the record to the end of the file.
- If the record pointer is on any record except the last record, WRITEfn overwrites the record following the record pointer (not the current record).
- The REWRITEfn command simplifies some writes to UX files. REWRITEfn is described in this section.
- When you write over an exiting record, the value of #RECLen must match the length of the existing record. If the record lengths do not match, an error message displays and the format aborts.
- WRITEfn appends a new-line character to each UX file record. For example, if #RECLen=80, the write operation takes 80 characters from the record buffer and adds a new-line character to the end before writing the data to a file.

Section 4
Basic Commands

Examples

WRITE6

Writes the contents of logical file 6's record buffer into file 6.

WRITE3,C1

Writes the contents of logical file 3's record buffer into file 3. If file 3 is a PR file, does a form feed (carriage control channel 1), then writes the record. (If file 3 is not a PR file, the C subcommand has no effect.)

WRITE1,BR

Raises a break condition on the AS line. Does not write the record buffer contents.

WRITE6(RECLEN=A3)

Writes the contents of logical file 6's record buffer into logical file 6. The number of characters written is determined by the value of accumulator 3.



Section 5
Programming Elements Used with Record Buffers

INTRODUCTION

This section describes two format programming elements used to manipulate data in record buffers. Table 5-1 lists these elements and describes what each one does.

Table 5-1. Programming Elements Used with Record Buffers

Element	Explanation
BLANKRfn	A format command that fills file <u>fn's</u> record buffer with blanks.
Rfn	A validation/generation descriptor that lets you access data in file <u>fn's</u> record buffer.

General Usage Notes

- MSFP automatically allocates a record buffer for each file you open. The record buffer's length is set as follows:
 - SD files: the buffer is equal to the maximum record length of the job specified in the \$DEFJOB supervisory command.
 - XD files: the buffer is equal to the maximum record length of the job specified in the \$INDSET supervisory command.
 - PR files: the buffer size is equal to the line length specified in the \$LENGTH parameter of the file type definition. If you omit \$LENGTH, the buffer size is equal to the line length specified in the V2config form. If no line length was specified in the V2config form, the line length is set to 132 characters.
 - AS and UX files: the buffer size is equal to the maximum record size specified in the file type definition. If no record size was specified, the buffer size is set to 1500 characters.
- If your format attempts to display a record buffer that contains non-displayable characters those characters are converted to tildes (~) for display purposes.

For example, assume that an AS file's record buffer starts with an STX character, then contains 12 displayable data characters, and ends with an NL character. The buffer would display like this:

~THIS IS DATA~

Section 5
Programming Elements Used with Record Buffers

THE BLANKRfn COMMAND

Use the BLANKRfn command to fill file fn's record buffer with blanks.

The syntax of the BLANKRfn command is:

BLANKRfn

where

fn is the logical file number of the file associated with the record buffer to be blank-filled.

Usage Notes

- Each record buffer is initialized to blanks when the associated file is opened.
- Use BLANKRfn to blank the record buffer after each WRITEfn when you are working with variable-length records.

Examples

BLANKR6

Blank-fills the record buffer associated with logical file 6.

THE Rfn VALIDATION/GENERATION DESCRIPTOR

Use SET statements with the Rfn descriptor to access the contents of file fn's record buffer. You can:

- Generate data from the record buffer.
- Validate against data in the record buffer.
- Set the value of any part of the record buffer.
- Use any part of the record buffer in an arithmetic expression.

The syntax of the Rfn descriptor is:

`Rfn(c,l)`

where

fn is the logical file number of the file associated with the record buffer being accessed.

c is the beginning column (1-9999) of the data being manipulated.

l is the length in columns (1-9999) of the data being manipulated.

Usage Notes

- The Rfn descriptor is valid on either side of a SET statement. See "Examples" below.
- You can reference the maximum column of a record buffer, even if the current record length is shorter than the buffer.
- The Rfn descriptor can be useful in setting and testing for non-displayable control characters in the record buffer for an AS or UX file. Used with the \hh descriptor, Rfn can set or test for the hexadecimal values that in represent such characters in the buffer. (See the LiFE-Works Format Language Manual, Volume 1 for additional details regarding the \hh descriptor.)
- For general information on using Rfn in arithmetic expressions, see the LiFE-Works Format Language Manual, Volume 1.

Section 5
Programming Elements Used with Record Buffers

Examples

```
SET R4(81,10)=R5(1,10)
```

Sets columns 81-90 of file 4's record buffer equal to columns 1-10 of file 5's record buffer.

```
IF R2(1,5)>"05000" THEN F2 ELSE F3
```

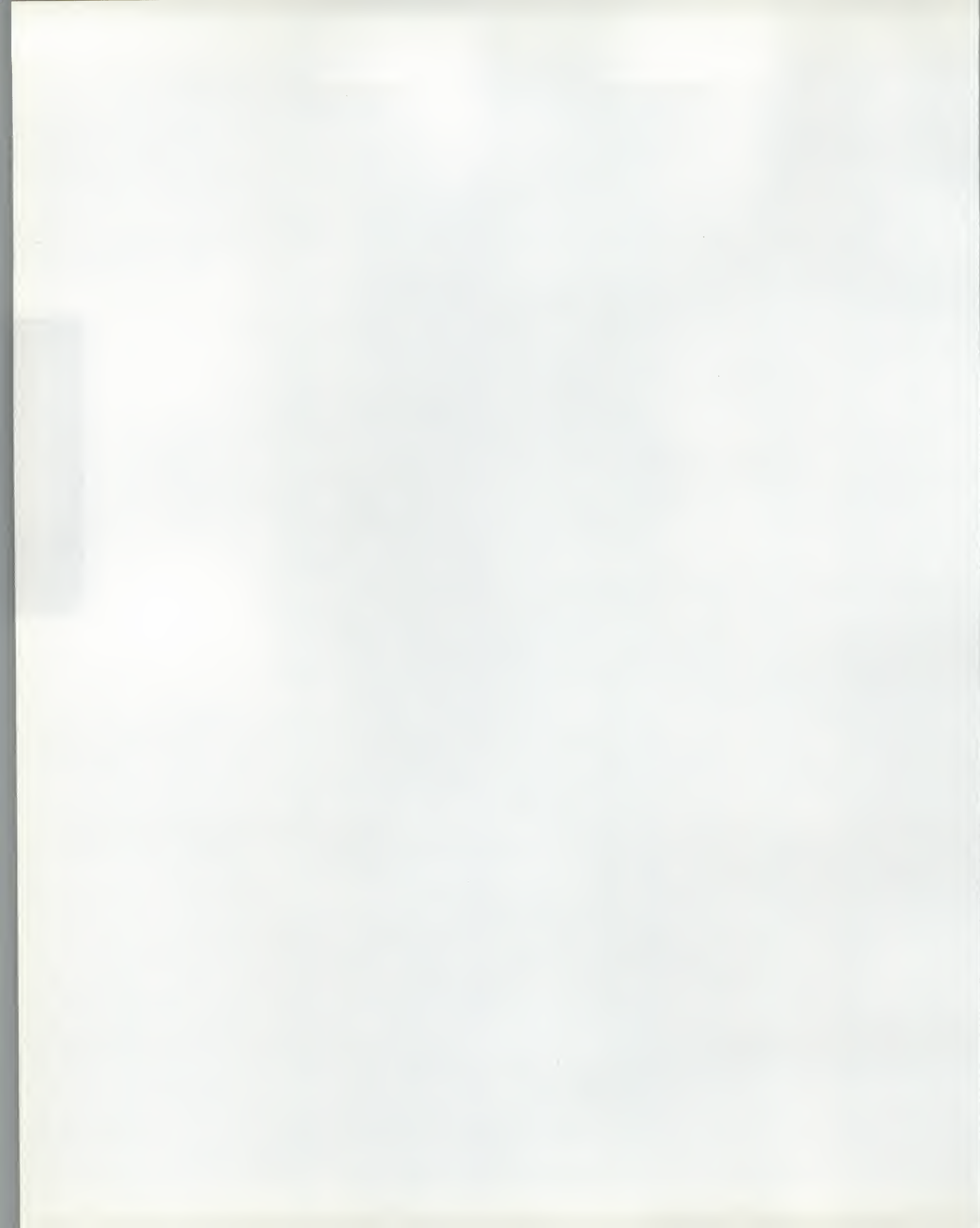
Validates the data in columns 1-5 of file 2's record buffer against the value 5000. If the data in the record buffer is greater than 5000, goes to program level 2. If the data in the record buffer is equal to or less than 5000, goes to program level 3.

```
SET R1(78,2)=\0DOA
```

Sets columns 78 and 79 of file 1's record buffer equal to hexadecimal value 0DOA, which represents a carriage return and line feed.

```
IF R5(1,1)=\02 THEN F2 ELSE F3
```

Checks the first character in file 5's record buffer. If that character equals hexadecimal 02 (an STX character), then goes to program level 2; otherwise goes to program level 3.



Section 6
Conditional Indicators

INTRODUCTION

This section discusses a set of conditional indicators used specifically in MSFP programming.

Conditional indicators are used in IF statements to test for a given condition. Table 6-1 lists the MSFP conditional indicators and explains what condition each indicator tests for.

Table 6-1. MSFP Conditional Indicators

Indicator	Explanation
BOBfn	Tests for beginning-of-batch in file <u>fn</u> . Applies only to SD files.
BOFfn	Tests for beginning-of-file in file <u>fn</u> .
EOBfn	Tests for end-of-batch in file <u>fn</u> . Applies only to SD files.
EOFfn	Tests for end-of-file in file <u>fn</u> .
OPENEDfn	Tests whether file <u>fn</u> is open.
OPENINfn	Tests whether file <u>fn</u> is open for input.
OPENOUTfn	Tests whether file <u>fn</u> is open for output.
RECSELfn	Tests whether a record is selected in file <u>fn</u> . Applies only to XD files.

Section 6
Conditional Indicators

THE BOBfn CONDITIONAL INDICATOR

Use the BOBfn conditional indicator to test for beginning-of-batch in file fn. BOBfn is designed to let you test for batch boundaries in SD files that are order chains.

Usage Notes

- BOBfn applies only to SD files. It is always set false for other file types.
- In SD files that are single batches, BOBfn is true whenever the record pointer is positioned before the first record in the file. This occurs when:
 - The file is first opened for input.
 - The file is first opened for output with the rewind [R] or clear [C] option.
 - An empty file is opened for output.
 - A LOCATEfn BOF is executed in the file.
 - You try to locate backward beyond the beginning of the file.
- In SD files that are order chains, BOBfn is true under all of the above conditions and whenever the record pointer is positioned before the first record of any batch in the order chain. This occurs when you try to locate backward beyond the beginning of any batch in the chain.
- In single-batch SD files, BOBfn is true whenever BOFfn is true and vice versa. In order-chain SD files, BOBfn is true whenever BOFfn is true but not vice versa.

Examples

1L2 LOCATE1 -F1 IF BOB1 THEN "ERROR--YOU CAN'T GO BACK TO THE PREVIOUS BATCH."

Reads a two-digit number entered on the screen and moves the record pointer backward for the specified number of records. Sends a message to the operator if the entered number would move the record pointer back beyond the beginning of a batch.

THE BOFfn CONDITIONAL INDICATOR

Use the BOFfn conditional indicator to test for beginning-of-file in file fn.

Usage Notes

- BOFfn applies to all file types except AS files. It is always set false for AS files.
- BOFfn is set true whenever the record pointer is positioned before the first record in a file. This occurs when:
 - The file is first opened for input.
 - The file is first opened for output with the rewind [R] or clear [C] option.
 - An empty file is opened for output.
 - A LOCATEfn BOF is executed on the file.
 - You try to locate backward beyond the beginning of a file.
- In an SD file that is an order chain, BOFfn is set true only when the record pointer is positioned before the first record in the first batch of the order chain. In other words, an order chain is treated as one big file.

The BOBfn and EOBfn conditional indicators let you test for batch boundaries within an order chain. Both are described in this section.

- BOFfn is not set true by a READfn(KEY...) command in an XD file.
- RECSELfn is always false when BOFfn is true.

Examples

```
IF BOF2 THEN READ2 ELSE (LOCATE2 BOF READ2)
```

If the record pointer is positioned at the beginning of logical file 2, then read the first record from file 2; if not, move to the beginning of file 2, then read the first record from file 2.

THE EOBfn CONDITIONAL INDICATOR

Use the EOBfn conditional indicator to test for end-of-batch in file fn. EOBfn is designed to let you test for batch boundaries in SD files that are order chains.

Usage Notes

- EOBfn applies only to SD files. It is always set false for other file types.
- In SD files that are single batches, EOBfn is set true when the record pointer is positioned after the last record in the file. This occurs when:
 - A file is first opened for output.
 - A LOCATEfn EOF is executed on the file.
 - A READfn is attempted after the last record in the file.
 - You try to locate forward beyond the end of the file.
- In SD files that are order chains, EOBfn is set true under all of the above conditions and each time a READfn reads the first record of a batch in the order chain.
- In single-batch SD files, EOBfn is true whenever EOFfn is true and vice versa. In order-chain SD files, EOBfn is true whenever EOFfn is true but not vice versa.

Examples

```
IF EOB6 THEN SET B(1,4)="0000" ELSE NULL
```

If the record pointer is at the end of a batch in logical file 6, place zeroes in columns 1 through 4 of the work buffer; if not, take no action.

THE EOFfn CONDITIONAL INDICATOR

Use the EOFfn conditional indicator to test for end-of-file in file fn.

Usage Notes

- EOFfn applies to all file types except AS files. It is always set false for AS files.
- EOFfn is true whenever the record pointer is positioned at the end of the last record in the file. This occurs when:
 - A file is first opened for output.
 - A LOCATEfn EOF is executed on the file.
 - You try to read after the last record in the file.
 - You try to locate forward beyond the end of the file.
- In an SD file that is an order chain, EOFfn is set true only when the record pointer is positioned at the end of the last record of the last batch in the order chain. In other words, an order chain is treated as one big file.

The BOBfn and EOBfn conditional indicators let you test for batch boundaries within an order chain. Both are described in this section.

- You should normally check the EOFfn command before each READfn command. READfn commands are ignored when EOFfn is true; that is, if EOFfn is true, READfn does not change the contents of the record buffer.

Examples

```
IF EOF2 THEN F3 ELSE READ2
```

If the record pointer is at the end of logical file 2, go to program level 3; if not, read the next record from file 2.

THE OPENEDfn CONDITIONAL INDICATOR

Use the OPENEDfn conditional indicator to test whether file fn is open.

Usage Notes

- OPENEDfn applies to all file types.
- OPENEDfn is set true if file fn is open. It stays true until the file is closed through CLOSEfn or the operator exits the program.

Examples

```
IF OPENED3 THEN READ3 ELSE F1
```

If logical file 3 is open, read the next record from file 3; if not, go to program level 1.

```
IF !OPENED2 THEN OPEN2,166,0 ELSE NULL
```

If logical file 2 is not open, assign logical file number 2 to Ftype 166 and open the file for output; if file 2 is already open, take no action.

THE OPENINfn CONDITIONAL INDICATOR

Use the OPENINfn conditional indicator to test whether file fn is open for input.

Usage Notes

- OPENINfn applies to all file types, but is always set false for PR files.
- OPENINfn is set true if the file being tested is open for input. It is set false if the file is open for output or is closed.

Examples

```
IF OPENED2 THEN (IF OPENIN2 THEN F4 ELSE F5) ELSE F1
```

If logical file 2 is open for input, goes to program level 4 (which might contain code for reading data without writing). If logical file 2 is open for output, goes to program level 5 (which might contain code for reading and writing data). If logical file 2 is not open, goes to program level 1 (which might contain open logic).

Section 6
Conditional Indicators

THE OPENOUTfn CONDITIONAL INDICATOR

Use the OPENOUTfn conditional indicator to test whether file fn is open for output.

Usage Notes

- OPENOUTfn applies to all file types.
- OPENOUTfn is set true if the file being tested is open for output. It is set false if the file is open for input or if the file is closed.

Examples

IF OPENOUT1 THEN (Z1 F2) ELSE F3

If logical file 1 is open for output, zero accumulator 1 and go to program level 2. If not, go to program level 3.

THE RECSELfn CONDITIONAL INDICATOR

Use the RECSELfn conditional indicator to determine if a previous READfn or INSERTfn command has selected a record from XD file fn.

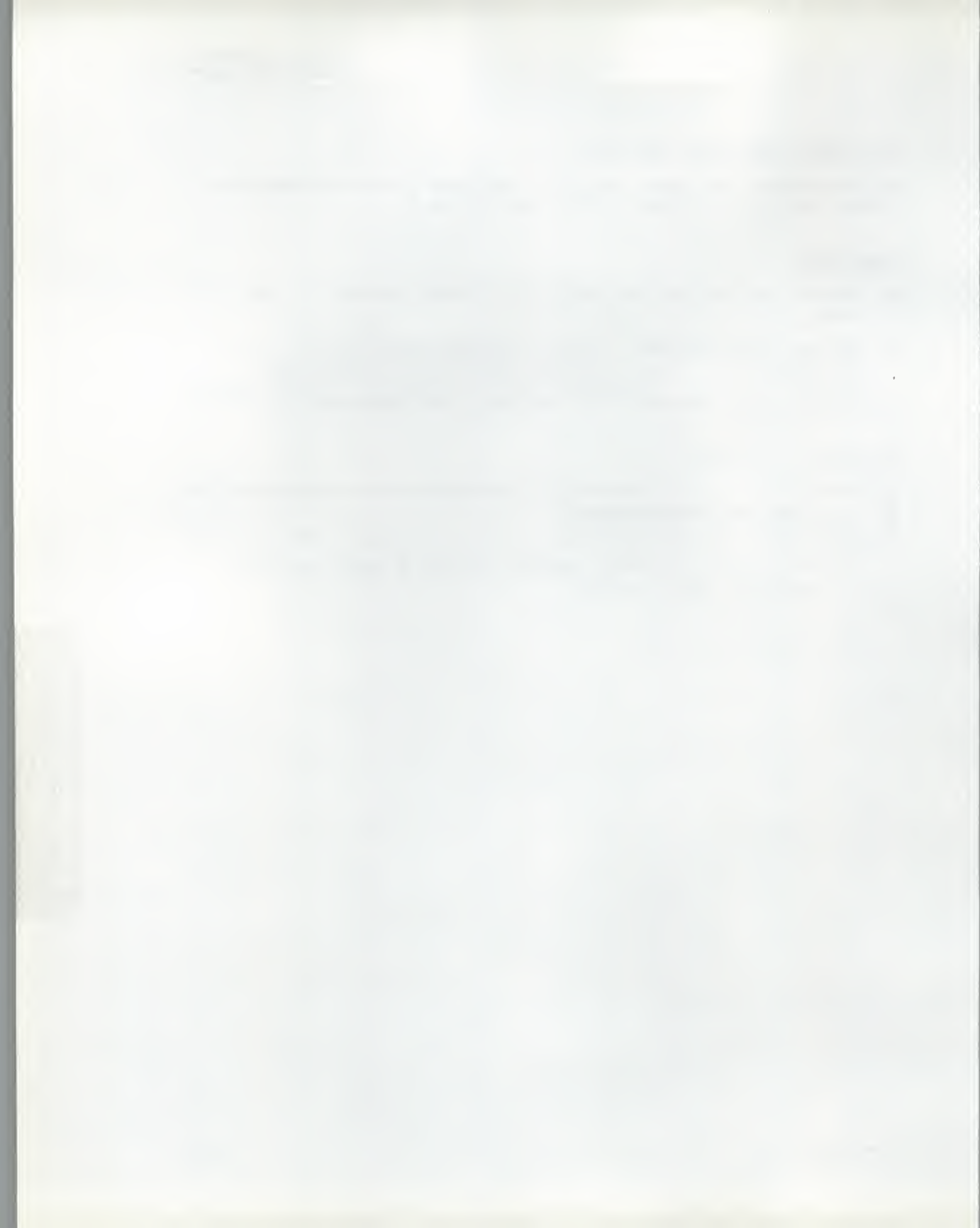
Usage Notes

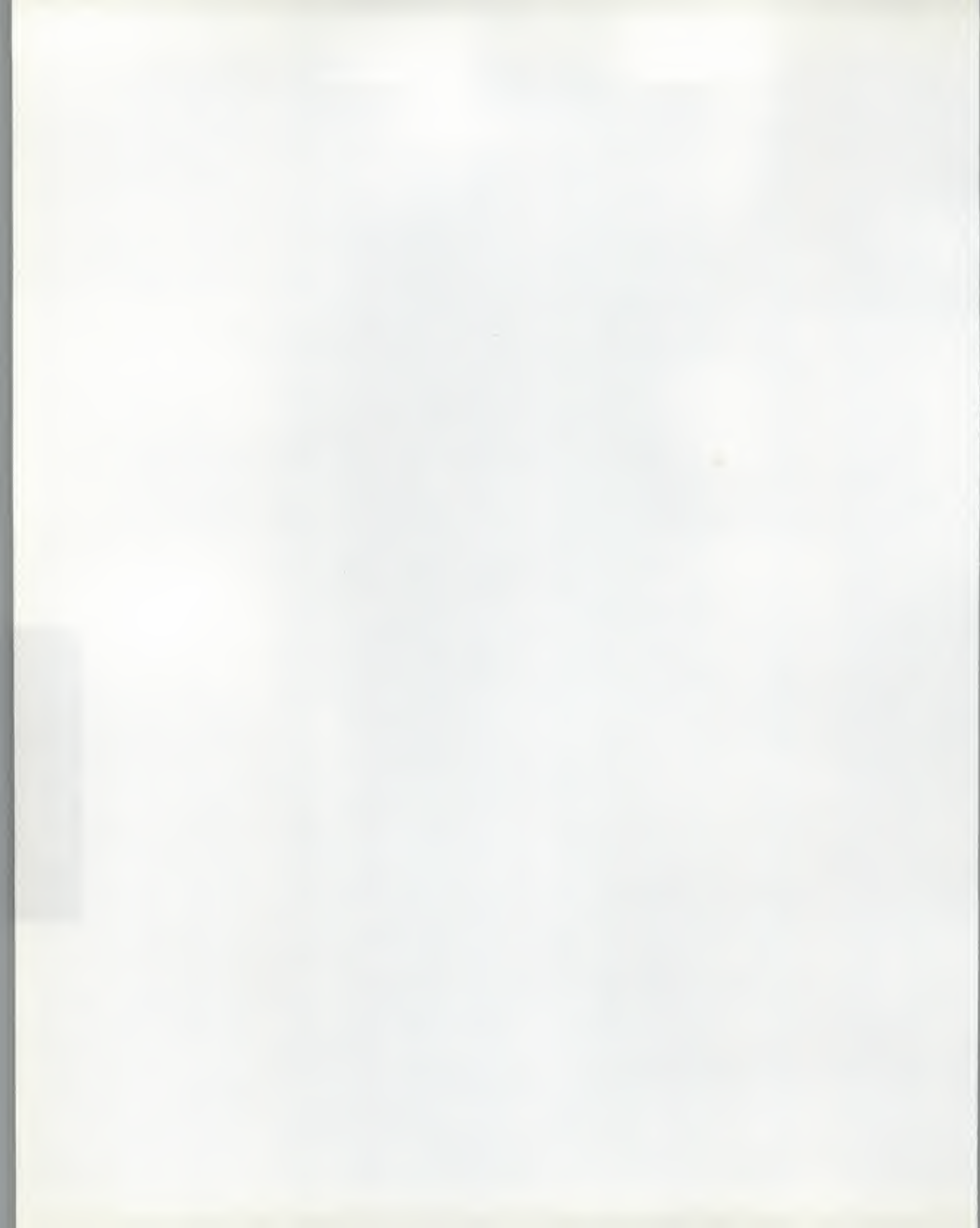
- RECSELfn applies only to XD files. It is always set false for other file types.
- RECSELfn is set true when a READfn or INSERTfn command selects a record from an XD file. The next LOCATEfn, DELETEDfn, or RELEASEfn command executed in the file sets RECSELfn false. A WRITEfn or REWRITEfn command does not change RECSELfn to false unless you use the release [R] parameter.

Examples

```
IF RECSEL4 THEN DELETE4 ELSE NULL
```

If a record is currently selected from file 4, delete the record; otherwise, take no action.





Section 7
Validation/Generation Descriptors

INTRODUCTION

This section describes a set of validation/generation descriptors used specifically in MSFP programming. These descriptors are keywords that take their values from the system. You can also set a value for some of these descriptors by using a SET statement.

Table 7-1 lists the MSFP validation/generation descriptors and explains what value each descriptor represents.

Table 7-1. MSFP Validation/Generation Descriptors

Descriptor	Explanation
#BATCHfn	Six-character batch identifier or spool file name associated with file <u>fn</u> .
#ERRORfn	Six-character descriptor that returns the error status of the node (system on OfficeLAN) or AS file associated with file <u>fn</u> . Used in conjunction with the RESULT command.
#FILEfn	Fourteen-character PR spool file name associated with file <u>fn</u> , or last component of the path name associated with AS or UX file <u>fn</u> .
#JOBfn	Nine-character job name associated with file <u>fn</u> .
#NODEfn	Twenty-character node name (name of the system on OfficeLAN) associated with file <u>fn</u> .
#PLEVELfn	Two-digit program level number of the last record read from file <u>fn</u> .
#RECLFNfn	Four-digit record length of the last record read from file <u>fn</u> .
#RECNUMfn	Six-digit record number of the last record read from file <u>fn</u> .

THE #BATCHfn SYSTEM KEYWORD

The #BATCHfn system keyword refers to the batch name or spool file name associated with file fn.

Usage Notes

- #BATCHfn is six characters long and is left-justified.
- #BATCHfn is set as follows:
 - SD files: #BATCHfn is set to the batch name specified in the file type definition for file fn. If the SD file is an order chain of batches, #BATCHfn is set to the batch in which the record pointer is currently positioned.
 - XD files: #BATCHfn is set to the batch specified in the \$IXBATCH parameter of the \$INDSET command.
 - PR files: #BATCHfn is set to the spool file name specified for the printer file. If the spool file name is longer than six characters, #BATCHfn is set to the first six characters of the name. (The system keyword #FILEfn, described in this section, lets you generate or validate a 14-character spool file name.)
- AS and UX files: #BATCHfn is set to blanks.
- #BATCHfn can be used only for validation or generation. You cannot set its value with a SET statement.

Examples

```
IF #BATCH6="FRIDAY" THEN CLOSE6 ELSE NULL
```

Closes logical file 6 if it is a batch named FRIDAY; otherwise, takes no action. (If logical file 6 is an order chain of SD batches, this command closes the file if the record pointer is currently in batch FRIDAY.)

```
ASD G6G#BATCH4
```

Generates the name of the batch associated with logical file 4 into a six-character general field.

THE #ERRORfn SYSTEM KEYWORD

The #ERRORfn system keyword returns error status. This error status may apply to either an AS file or a node (system on OfficeLAN) associated with file fn. You must use the RESULT command to turn on format control over error processing before you use #ERRORfn. See "Usage Notes" below for details.

Usage Notes

- #ERRORfn is six characters long and is left-justified.
- #ERRORfn contains the value of the message returned from certain MSFP OfficeLAN and AS file operations. #ERRORfn returns one of the following values:

000000	Successful completion. #ERRORfn is set to 000000 after a successful OPENfn or when used with local files.
020001	The specified host does not exist.
020002	Life-Works is not available on the specified host.
020003	All OfficeLAN circuits are in use.
020004	Contact with remote system lost.
011500	Timeout occurred on AS file open, read, or write.
011501	Break condition occurred on the AS line, or the operator aborted the format.
- #ERRORfn is used in conjunction with the RESULT command. RESULT turns on format control over MSFP error processing. You can insert the RESULT command anywhere in your format code, as long as you turn it on before you use #ERRORfn. If you do not turn on RESULT, and you open an AS file or a file on another system on OfficeLAN, then errors that occur will cause your format program to abort.

RESULT allows your format program to continue processing even if one of the specified files has aborted because of one of the above errors. Consequently, your program should provide for the possibility that Life-Works can encounter a system-level error, and should avoid accessing an aborted file until the problem is fixed and the file is reopened.

For example, if a system on OfficeLAN crashes while MSFP is being used across the LAN, then all the files on that system will return errors when the format code tries to access them. When the format code detects this situation by checking #ERRORfn, it should provide for closing those files and bypassing attempts to access them until the crashed system is brought back up.

Section 7
Validation/Generation Descriptors

- With RESULT on for an AS file, any of the above errors occurring during OPENfn, CLOSEfn, READfn, or WRITEfn will not abort the format.
- With RESULT on, the format code will still abort when encountering general MSFP errors.
- To turn RESULT off, use the RESULT# command. When RESULT is off (#RESULT), any error will abort the format.
- #ERRORfn can be used only for validation or generation. You can't set its value with a SET statement.

Examples

```
RESULT
.
.
.
WRITE4
IF #ERROR4!"000000"
THEN
(
    SET R5(1,80)=R4(1,80)
    WRITE5
)
ELSE
    NULL
```

Logical file 4 is a remote file. Logical file 5 is a local file. An attempt is made to write a record to file 4. If the write is not successful (if #ERROR4 is not equal to "000000"), the record is copied into file 5's record buffer and written to a file locally.

THE #FILEfn SYSTEM KEYWORD

The #FILEfn system keyword refers to either the spool file name associated with PR file fn or the file name of AS or UX file fn.

Usage Notes

- #FILEfn is similar to #BATCHfn but is designed specifically for PR, AS, and UX files. #FILEfn can accommodate a complete spool file name (up to 14 characters); #BATCHfn can accommodate only the first six characters of a spool file name.
- #FILEfn is 14 characters long and is left-justified.
- #FILEfn is set as follows:
 - SD and XD files: #FILEfn is set to blanks.
 - PR files: #FILEfn is set to the name of the spool file (14 characters maximum).
 - AS and UX files: #FILEfn is set to contain the last component of the path name to the file.
- #FILEfn can be used only for validation or generation. You cannot set its value with a SET statement.

Examples

ASD G14G#FILE2

Generates the name of the spool file associated with logical file 2 into a 14-character general field.

Section 7
Validation/Generation Descriptors

THE #JOBfn SYSTEM KEYWORD

The #JOBfn system keyword refers to the job name associated with the SD or XD file fn.

Usage Notes

- #JOBfn is nine characters long and is left-justified.
- #JOBfn is set as follows:
 - SD files: #JOBfn is set to the job name specified in the file type definition for file fn.
 - XD files: #JOBfn is set to the job name specified in the \$IXBATCH parameter of the \$INDSET command.
 - PR, AS, and UX files: #JOBfn is set to blanks.
- #JOBfn can be used only for validation or generation. You cannot set its value with a SET statement.

Examples

```
IF #JOB2="PAY" THEN READ2 ELSE CLOSE2
```

Reads a record from logical file 2 if the job name for file 2 is "PAY"; if not, closes logical file 2.

THE #NODEfn SYSTEM KEYWORD

The #NODEfn system keyword refers to the node name associated with file fn. If your system uses OfficeLAN, and fn is a file on another system in the OfficeLAN network, #NODEfn is the name of the system where fn resides.

Usage Notes

- #NODEfn is 20 characters long and is left-justified.
- #NODEfn can be used only for validation or generation. You cannot set its value with a SET statement.
- #NODEfn is blank if file fn is not defined as a remote ftype.
- #NODEfn is not applicable to AS and UX files.

Examples

```
ASD G20G#NODE14
```

Generates the name of the system where logical file 14 resides into a 20-character general field.

```
SET S(2,55,20)=#NODE2
```

Sets screen positions 55-75 to the current node name.

Section 7
Validation/Generation Descriptors

THE #PLEVELfn SYSTEM KEYWORD

The #PLEVELfn system keyword refers to the current program level of the SD or XD file fn.

Usage Notes

- #PLEVELfn is two digits long and is right-justified with leading zeros.
- #PLEVELfn is initialized to 01 when a file is opened. Each time a record is read, #PLEVELfn is set to that record's program level.
- When a record is written to an SD or XD file (WRITEfn, REWRITEfn, or INSERTfn), the current value of #PLEVELfn determines the output program level.
- #PLEVELfn is ignored for PR, AS, and UX files.
- You can set the value of #PLEVELfn with a SET statement.

Examples

```
SET #PLEVEL6=2
```

Sets #PLEVEL for logical file 6 to 2.

```
IF #PLEVEL6!2 THEN SET #PLEVEL6=2 ELSE NULL
```

If the current value of #PLEVEL for logical file 6 is not 2, sets it to 2; otherwise, takes no action.

THE #RECLFN SYSTEM KEYWORD

The #RECLFN system keyword refers to the length of the record that has just been read from file fn or written to file fn.

Usage Notes

- #RECLFN is four digits long and is right-justified with leading zeros.
- #RECLFN is set as follows:
 - SD and XD files: When the file is opened, #RECLFN is initialized to the maximum record length. Each time a record is read, #RECLFN is set to the actual record length.
 - PR files: #RECLFN is initialized to the length specified in the \$LENGTH parameter of the \$FTYPE or FTYPEfn command. (If \$LENGTH is not used, #RECLFN is set to the line length specified in the V2config form. If no line length was specified in the V2config form, the line length is set to 132 characters.)
 - AS and UX files: #RECLFN is initialized to 80. Each time a record is read, #RECLFN is set to the number of characters just read.
- When data is written into a file with variable-length records, the current value of #RECLFN determines the output record length.

Note that if an existing record is overwritten, the value of #RECLFN must equal the length of the record being overwritten. In other words, record-length changes are not permitted for existing records.
- You can set the value of #RECLFN with a SET statement. Setting the value of #RECLFN has no effect on the length of a printed line nor on the record length of a fixed-length SD or XD file.
- If the (RECLFN=n) option is used with a READfn or WRITEfn command for an AS file, that value takes precedence over #RECLFN for the read or write operation.

Examples

```
SET #RECLN3=80
```

Sets the value of #RECLFN for logical file 3 to 80.

Section 7
Validation/Generation Descriptors

IF #RECLN4<132 THEN F2 ELSE F3

If the current length of #RECLN for logical file 4 is less than 132 characters, go to program level 2; if not, go to program level 3.

THE #RECNUMfn SYSTEM KEYWORD

The #RECNUMfn system keyword reflects the number of the record just read from file fn or written to file fn.

Usage Notes

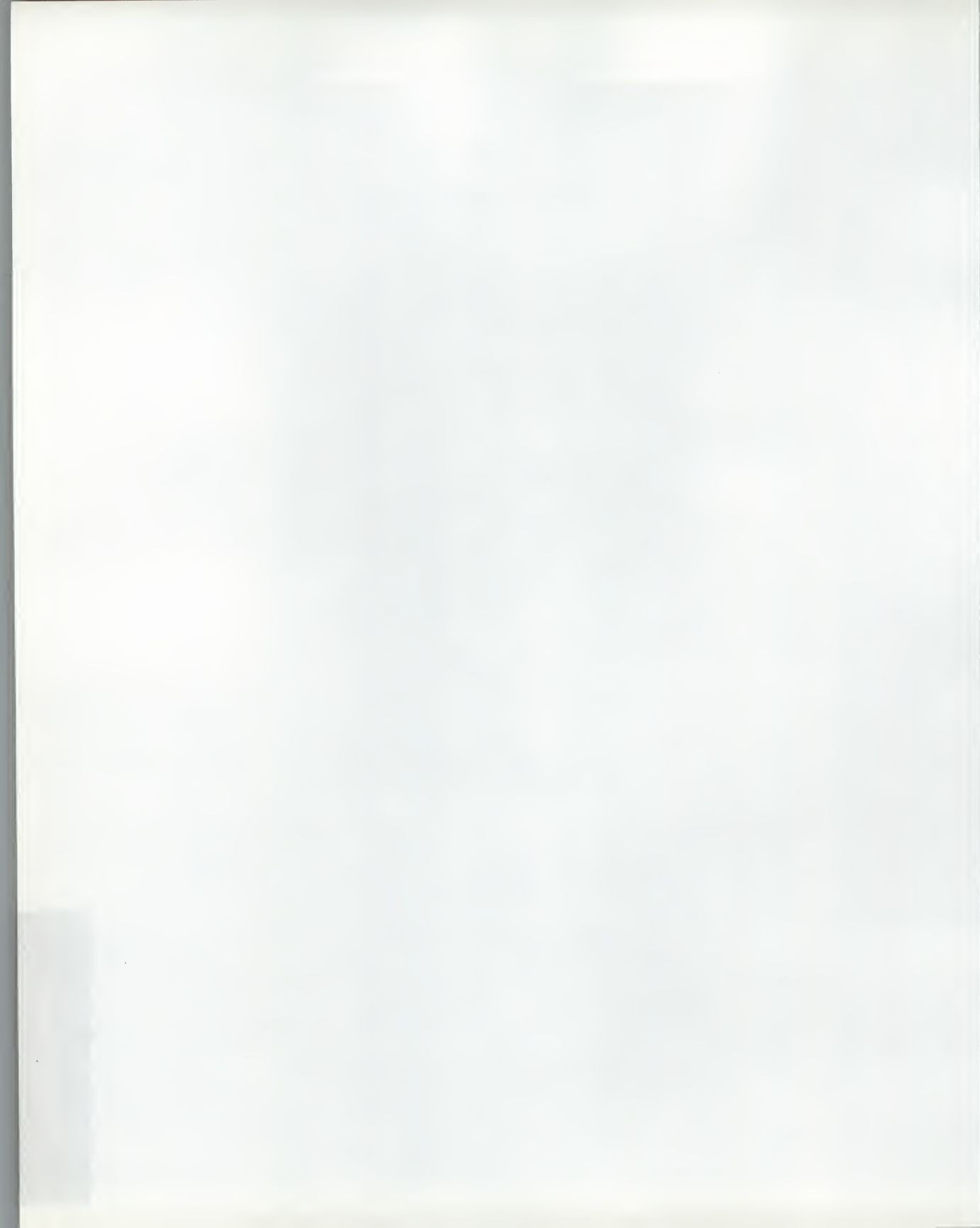
- #RECNUMfn is a six-digit number and is right-justified with leading zeros.
- #RECNUMfn is set as follows:
 - SD, UX, and PR files: If the file is opened at beginning-of-file, #RECNUMfn is initialized to zero. If the file is opened at end-of-file, #RECNUMfn is initialized to the total number of records in the file. Each READfn or WRITEfn command sets #RECNUMfn equal to the record number of the record read or written.
 - XD files: #RECNUMfn is always initialized to zero, regardless of whether the file is opened at beginning-of-file or end-of-file. Each READfn(KEY=...) or READfn(KEY>=...) command sets #RECNUMfn to 50,000. Each READfn command (sequential read) increments #RECNUMfn by 1.
 - AS files: #RECNUMfn is initialized to zero. Each READfn or WRITEfn command increments #RECNUM by one.
- The LOCATEfn command does not affect the value of #RECNUMfn.
- #RECNUMfn can be used only for validation or generation. You cannot set its value with a SET statement.

Examples

```
IF #RECNUM5>1000 THEN CLOSE5 ELSE WRITE5
```

If the current value of #RECNUM for logical file 5 is greater than 1000, then close logical file 5; otherwise, write a record to logical file 5.





Section 8 MSFP Printing

The information in this section assumes that you are doing LiFE-Works printing through the Motorola Print System. If you are doing LiFE-Works printing with the traditional operating system (lp) facility, see Appendix B, "MSFP Printing with lp." Refer to the LiFE-Works Supervisor's Manual for further information on LiFE-Works printing.

ABOUT MSFP PRINTING

MSFP lets you do several print operations that can't be done through other kinds of LiFE-Works printing. When you print through MSFP you can:

- Access a printer directly from a format program.
- Select and print individual records from a batch.
- Precisely control the vertical movement of paper through the printer.

This section describes printing procedures that are specific to MSFP. This section assumes that you are familiar with LiFE-Works printing as described in the LiFE-Works Supervisor's Manual.

This section covers the following topics:

- Using a Vertical Format Unit (VFU). LiFE-Works uses simulated VFUs to move the paper through the printer.
- Defining MSFP forms. An MSFP form definition lets you define a specific printer/Print System form combination as a PR file and refer to it in an MSFP program.
- Using spool files. MSFP prints by sending data to a Print System spool file, which the Print System then sends to the printer. You can choose between permanent or temporary spool files, and you can name the spool file yourself or have the system generate a name.
- Using the SPOOL FILE DISPLAY. The SPOOL FILE DISPLAY lists permanent spool files and lets you print or delete listed files.

USING A VERTICAL FORMAT UNIT (VFU)

A vertical format unit (VFU) is a device that controls the vertical movement of paper through a printer. LiFE-Works uses VFU simulation; that is, it uses software instructions to accomplish the functions of a physical VFU device.

You don't need to know much about physical VFU devices to print through MSFP, but you do need to understand the basic concepts and terminology. If you are not familiar with VFUs, refer to the Motorola Print System User's Manual. If you are familiar with VFUs, just keep in mind that for LiFE-Works, "channels," "punches," and so on refer to software instructions rather than parts of a device.

LiFE-Works uses the Print System's VFU simulation for printer control. However, nonMSFP LiFE-Works printing uses only carriage control channels 1 and 2. (Typically, channel 1 makes the paper advance to top-of-form, and channel 2 makes the paper advance one line at a time.) MSFP lets you take full advantage of a VFU. Using the MSFP WRITEfn command, you can:

- Access any of the 12 VFU channels (WRITEfn,Cc).
- Make the paper advance a specified number of lines (WRITEfn,Ss).
- Prevent the paper from moving, thus causing overprinting (WRITEfn,CR).

The Print System contains a default VFU definition that should satisfy most of your VFU requirements. If you want to create your own VFU definition, or you want more information on the default VFU definition, refer to the Motorola Print System User's Manual.

DEFINING MSFP FORMS

An MSFP form lets you include the following information as part of an MSFP program:

- The name of the printer to be used
- The Motorola Print System form to be used
- A heading to be printed at the top of each page

To define an MSFP form, enter and execute the following supervisory commands:

```
$FORMS vvv
```

```
[F=form] [P=printer] [H=heading]
```

where

vvv is a unique three-digit identifier (000-999).

printer is the name of the printer to be used. If you omit the P parameter, MSFP uses whatever printer is currently assigned to the terminal running the MSFP format program.

form is the Motorola Print System form to be used. If you omit the F parameter, MSFP uses whatever Print System form is currently assigned to the terminal running the MSFP format program.

heading generates a heading at the top of each page. Use the following commands to generate a heading:

Tw,text generates a text string w characters long, where text is the text string.

Sw skips (blanks) w columns.

Gw,c generates w repetitions of the character c. c can be any character except a blank.

P generates sequential page numbers. Page numbers start at 1 and can go up to 99999. Page numbers are right-justified within the possible five character positions.

#keyword generates the current value of a system keyword. keyword can be any LIFE-Works system keyword except those used specifically with MSFP.

/ causes a line feed.

The heading can contain up to a total of 190 characters, including skipped columns and line feeds.

Section 8
MSFP Printing

Usage Notes

- To use an MSFP form in a program, you refer to the form in the \$FORMS vvv parameter of a PR file type definition (where vvv is the three-digit MSFP form identifier). If you don't specify an MSFP form, print output from your program is sent to whatever printer/Print System form is currently assigned to the terminal running the MSFP format program.
- MSFP forms are system elements, like formats and value sets. They can be used by any number of MSFP programs. Select mode S-D to see a list of the MSFP forms on your system.
- \$FORMS vvv must be on a record by itself. The remaining \$FORMS parameters can extend over multiple records (but individual parameters can't be split).
- You can use separating commas rather than blanks between the \$FORMS parameters. (See "Examples" below.)
- Generating a heading in an MSFP form is much like generating a heading in disk-to-printer reformatting. If you're not familiar with this procedure, see the LIFE-Works Supervisor's Manual.
- For information on defining Print System forms, refer to the Motorola Print System User's Manual.

Examples

```
$FORMS 111
F=INVOICE P=CP1-INVOICE
```

Defines printer CP1-INVOICE and Print System form INVOICE as MSFP form 111.

```
$FORMS 200
F=REPORT,P=CP2-REPORT
H=S20 T48,COPY CENTER CROSS-CHARGES (SORTED BY DEPARTMENT) S5 PAGE: P
```

Defines printer CP2-REPORT and Print System form REPORT as MSFP form 200, generates a page heading, and numbers the pages sequentially.

USING SPOOL FILES

Each MSFP PR file (printer) has a spool file associated with it. When a format program writes data to a PR file, the spool file receives the data. When the program closes the PR file, the associated spool file is placed in the print queue.

For each PR file, you can specify either a permanent spool file or a temporary spool file.

Permanent spool files remain on the system after being printed. You don't have to print a permanent spool file when you close it; you can delay printing by closing the file with the delay option (CLOSEfn,D).

Temporary spool files are deleted after printing. You cannot delay printing of a temporary spool file.

Use the S option of the OPENfn command to specify a permanent spool file, or use the T option to specify a temporary spool file. If you don't specify either S or T, MSFP uses a temporary spool file with a system-generated name.

NOTE

If you write a program that opens and closes the same PR file multiple times, your choice of spool files affects the way output data is handled. If you use a permanent spool file, the system appends output data to the end of the spool file each time you close the PR file. (Page numbering doesn't start over with each addition.) If you use a temporary spool file, the system sends the data to a new spool file each time you close the PR file.

You have a number of choices for naming spool files. You can:

- Supply a spool file name in the \$SPOOL parameter of the \$FTYPE or FTYPEfn command:

\$FTYPE...\$SPOOL spoolfile

- Supply a spool file name in the OPENfn command:

$$\text{OPENfn} \dots \left\{ \begin{array}{c} \text{S} \\ \text{T} \end{array} \right\} = \text{"spoolfile"}$$

Section 8
MSFP Printing

- Generate a spool file name into the OPENfn command from another source, such as an accumulator or buffer:

$$\text{OPENfn} \dots \left\{ \begin{array}{c} \text{S} \\ \text{T} \end{array} \right\} = \text{reference}$$

- Have the spool file name entered at run time:

\$FTYPE ...\$SPOOL *

- Have the system generate the spool file name:

$$\text{OPENfn} \dots \left\{ \begin{array}{c} \text{S} \\ \text{T} \end{array} \right\} \quad \text{or} \quad \text{OPENfn with no spool file option--generates a temporary spool file}$$

Spool file names can be 1-14 characters long. They can contain any characters except slashes (/) and backslashes (\), but the first character must not be a period.

System-generated permanent spool file names have the form

Vtttffnnn

where

ttt is the current terminal number.

ff is the logical file number assigned to the PR file.

nnn is the form number if \$FORMS was included in the \$FTYPE or FTYPEfn command. If not, nnn is blank.

System-generated temporary spool file names have the form

PSxxxxx

where xxxxx is a sequential five digit number.

If you use more than one naming method in the same program, LIFE-Works applies the following precedence:

1. The name entered at run time (highest precedence).
2. The name specified in (or generated into) the OPENfn command.
3. The name specified in the \$FTYPE or FTYPEfn command.
4. The system-generated name (lowest precedence).

USING THE SPOOL FILE DISPLAY (MODE P-S)

The SPOOL FILE DISPLAY lists the name of every spool file on the system. The files in this display are usually permanent spool files, but temporary spool files also appear while they are in a print queue. You can reach the SPOOL FILE DISPLAY by selecting Mode P-S or by using the LiFE-Works menu system. Refer to the LiFE-Works Menu Manual for information on using the LiFE-Works menu system to reach this display. Figure 8-1 shows the SPOOL FILE DISPLAY.

SPOOL FILE DISPLAY				
PS00010	PS00011	PS00012	PS00013	PS00014
PS00015	PS00016	PS00017	PS00018	PS00019
PS00020	PS00021	V00169	PS00022	PS00023
PS00024	V00235411	spoolfile	PS00025	V00169
stardust	PS00026			

File detail _____ Print file _____ Delete file _____ Print, then delete _____	Enter file name and press <ACCEPT> <EXIT> return to main print menu <TOP MENU> return to top menu <HELP> display help
---	---

Figure 8-1. The SPOOL FILE DISPLAY

If there are too many spool files to display on one screen, use DOC FORWARD to display the next screen of spool files. Use DOC BACK to display the previous screen of spool files.

Options

- File detail shows the FILE DETAIL display. See the "File Detail Display" later in this section for more information.
- Print file prints the spool file using the Number of copies and the Destination Printer parameters of the FILE DETAIL display.
- Delete file deletes the spool file.
- Print, then delete prints and then deletes the spool file.

To select an option, type the name of the desired spool file in the space next to the option and press ACCEPT. If 2 or more spool files share the name you enter, LiFE-Works displays the DUPLICATE SPOOL FILE NAME SELECTION screen. See "Duplicate Spool File Name Selection Screen" below for further information on this screen.

Be sure that you enter a file name next to only one option. LiFE-Works treats the first filled option as the one you want even if you filled in others.

You can also use the command line to perform the functions of the SPOOL FILE DISPLAY. Select mode P-S and enter the name of the desired spool file. LiFE-Works shows the FILE DETAIL display. From there, you can print or delete spool files. See "Using the File Detail Display" below for instructions on printing and deleting spool files from the command line.

Duplicate Spool File Name Selection Screen

The DUPLICATE SPOOL FILE NAME SELECTION screen appears when you try to print, to delete, or to display detailed information on a spool file that has the same name as another spool file. This screen displays the full pathnames of the files, allowing you to identify the file you want. Figure 8-2 shows the THE DUPLICATE SPOOL FILE NAME SELECTION screen.

DUPLICATE SPOOL FILE NAME SELECTION	
<pre>/usr/spool/mps/psdir/spool/V00169 /usr/dlb/up/V00169</pre>	
	<pre>Position cursor and press <SELCT> <EXIT> return to main print menu <TOP MENU> return to top menu <HELP> display help</pre>

Figure 8-2. The DUPLICATE SPOOL FILE NAME SELECTION Screen

To perform the option you selected in the SPOOL FILE DISPLAY, position the cursor next to the desired file and press SELCT.

Using the File Detail Display

SPOOL FILE DISPLAY									
FILE DETAIL File name: PS00019 ① Last accessed with form: standard ② Last accessed by: david 10/09 10:28 ③ Number of copies: 1 ④ Number of pages: 15 ⑤ Destination Printer: cpr1 ⑥	<table style="width: 100%; border: none;"><tr><td style="width: 50%;">PS00013</td><td style="width: 50%;">PS00014</td></tr><tr><td>PS00018</td><td>PS00019</td></tr><tr><td>PS00022</td><td>PS00023</td></tr><tr><td>PS00025</td><td>V00169</td></tr></table>	PS00013	PS00014	PS00018	PS00019	PS00022	PS00023	PS00025	V00169
PS00013	PS00014								
PS00018	PS00019								
PS00022	PS00023								
PS00025	V00169								

- Print this file
- Delete this file
- First print, then delete this file

Position cursor and press <SELECT>
<EXIT> return to main print menu
<TOP MENU> return to top menu
<HELP> display help

Figure 8-3. The FILE DETAIL Display

Explanation

- ① The spool file name.
- ② The Print System form assigned to the file when it was created. If this field is blank, the file prints with the form that is assigned to the printer.
- ③ The operating system login name under which the spool file was created, followed by the date and time the file was created.
- ④ The number of copies to be printed.

- ⑤ The total number of pages in the spool file. The Print System calculates this number by dividing the print length (number of lines of text per page) of the file by the print length specified in the Print System form assigned to the file. If there is no Print System form assigned to the spool file, the Print System uses the print length from the default form MPS_DEF_F.
- ⑥ The printer on which the file is to be printed.

Options

- Print this file prints the spool file.
- Delete this file deletes the spool file.
- First print, then delete this file prints and then deletes the spool file.

You can also use the command line to print and delete spool files while you are in the FILE DETAIL display. To print a permanent spool file using mode P-S:

- 1 Press P to print the spool file, or press Q to print and delete the file.
- 2 Enter the name of the printer you want to use, then press RETURN; or press RETURN without entering anything to use the printer named in the display.

To delete a permanent spool file using mode P-S, press D.

The first part of the paper discusses the importance of the study of the history of the United States. It is argued that a knowledge of the past is essential for a full understanding of the present. The author then goes on to discuss the various factors which have shaped the development of the United States, including the influence of the British, the Spanish, and the French. He also discusses the role of the American people in the creation of the nation. The paper concludes by stating that the study of the history of the United States is a task of great importance, and that it is one which should be undertaken by all who are interested in the future of the country.



Appendix A
Sample Programs

EXAMPLE 1: MAINTAINING AN INDEX SET

The program on the following page maintains insurance claims records kept in an index set. The program checks each record in the index set to determine if the claim has been closed. If the claim is closed, the program prints the claims record and deletes it from the index set.

In this example:

- The job CLAIMS provides the operator interface. CLAIMS is not an MSFP file. It prompts the operator and keeps record counts.
 - Program level 1 of CLAIMS (format 201) opens the files and starts processing.
 - Program level 2 (format 202) does the processing.
 - Program level 3 (format 203) closes the files and stops processing.
 - Accumulator 1 counts the closed claims.
 - Accumulator 2 counts the total number of records read.
- Index set 100 contains the claims records. Index set 100 is defined as Ftype 001 (an XD file) and is logical file 1 in the code.
- Logical file 2 is a printer.

Appendix A
Sample Programs

Example 1:

```
$FTYPE 001,XD $INDSET 100
$FTYPE 002,PR $FORMS 200
$DEFJOB CLAIMS $FORMAT 201,202,203 $MAXSIZ 750 $ACCUMS 2

$DELETE FORMAT 201
$FORMAT 201 /* PROGRAM LEVEL 1--OPENS FILES & STARTS THE PROCESSING */
B1,20 P(H)"DELETE CLOSED RECORDS FROM MASTER FILE"
B3,20 P(H)"TYPE Y TO BEGIN PROCESSING, PRESS MODE TO ABORT"
E22,80
A(L)1E="Y" NDR
OPEN1,001,O,R
OPEN2,002,O,S="CLOSEDCLAIMS"
SET R2(20,39)="CLOSED RECORDS DELETED FROM MASTER FILE"
WRITE2,C1
Z1 Z2 F2

$DELETE FORMAT 202
$FORMAT 202 /* PROGRAM LEVEL 2--DOES THE PROCESSING */
@3,1 E3,80 NDR
P(H)"JOB IN PROCESS"
IF OPENED1
THEN
  (IF EOF1
    THEN F3
    ELSE (READ1
          SET A2=A2+1
          SET S(3,20,7)=A2
          IF R1(126,1)="C"
          THEN (SET R2(1,126)=R1(1,126)
                SET A1=A1+1
                WRITE2,S1
                BLANKR2 DELETE1)
          ELSE F2) )
ELSE F1

$DELETE FORMAT 203
$FORMAT 203 /* PROGRAM LEVEL 3--CLOSES FILES & ENDS PROCESSING */
@3,1 E3,80
SET R2(10,22)="TOTAL RECORDS DELETED"
SET R2(33,12)=A1
WRITE2,S2
CLOSE1 CLOSE2
NDR P(H)"PROGRAM COMPLETE" G(L)1
```


Explanation of Example 1:

\$FTYPE 001,XD \$INDSET 100

Create an XD file type definition for index set 100. Assign the identifier 001 to the XD file.

\$FTYPE 002,PR \$FORMS 200

Create a PR file type definition. Assign the identifier 002 to the PR file. The printer/form combination to be used is identified in MSFP form 200, which has already been defined.

\$DEFJOB CLAIMS \$FORMAT 201,202,203 \$MAXSIZ 750 \$ACCUMS 2

Define the job CLAIMS.

\$DELETE FORMAT 201

\$FORMAT 201 /* PROGRAM LEVEL 1--OPENS FILES AND STARTS THE PROCESSING */

B1,20 P(H)"DELETE CLOSED RECORDS FROM MASTER FILE

Blank the screen to row 1 column 20 and display an informative prompt for the operator.

B3,20 P(H)"TYPE Y TO BEGIN PROCESSING, PRESS MODE TO ABORT"

Blank the screen to row 3 column 20, and prompt the operator to begin processing.

E22,80

Erase the rest of the screen.

A(L)1E="Y" NDR

Define a one-character uppercase must-enter alphabetic field. When the operator enters Y, processing begins. (If the operator enters anything other than Y, the message NOT VALID DATA appears.) This record will not be saved (NDR).

OPEN1,001,0,R

Assign Ftype 001 as logical file 1, open the file for output, and rewind the file (move the record pointer to beginning-of-file. Ftype 001 is index set 100, which contains all claims records.

OPEN2,002,0,S="CLOSEDCLAIMS"

Assign Ftype 002 as logical file 2, open the file for output, and specify that output is to be placed in the permanent spool file CLOSEDCLAIMS. Ftype 002 is a PR file (printer).

SET R2(20,39)="CLOSED RECORDS DELETED FROM MASTER FILE"

Set file 2's record buffer equal to the string enclosed in quotes, beginning in column 20, for 39 characters. This string is printed as a report title.

Appendix A
Sample Programs

```
WRITE2,C1
    Do one form feed, and print record buffer 2.

Z1    Z2
    Zero accumulators 1 and 2 before processing begins.

F2
    Go to program level 2 (F2).

$DELETE FORMAT 202

$FORMAT 202          /* PROGRAM LEVEL 2*--DOES THE PROCESSING */

@3,1
    Move the cursor to row 3 column 1 (without erasing the DELETE CLOSED
    RECORDS prompt from program level 1).

E3,80    NDR
    Erase the screen to row 3 column 80.  NDR is used because this program
    level is for processing only.

P(H)"JOB IN PROCESS"
    Place a prompt at row 3, column 3, to inform the operator that processing
    is now taking place.

IF OPENED1 THEN
    Test to make sure file 1 is open.

(IF EOF1 THEN F3
    Branch to program level 3 if end-of-file has been reached in file 1.

ELSE (READ1
    If the file is not at the end, read a record from file 1 into record buffer
    1.

SET A2=A2+1  SET S(3,20,7)=A2
    Increment accumulator 2 by one to keep a running total of the records
    read.  Display the total on the screen at row 3 column 20.

IF R1(126,1)="C"
    Check the data in the buffer to see if the claim has been closed as
    indicated by a "C" in column 126 in record buffer 1.

THEN (SET R2(1,126)=R1(1,126)
    If the record is a closed claim, place it in record buffer 2.

SET A1=A1+1
    Increment accumulator 1 by one to keep a running total of the number of
    closed claims.

WRITE2,S1
    Do a line feed, then print the contents of record buffer 2.
```

BLANKR2 DELETE1)

Clear record buffer 2 for the next record, and delete the record containing the closed claim from file 1.

ELSE F2))

If the claim has not been closed, it is not deleted from file 1 and the program loops back to the beginning of program level 2 to read the next record.

ELSE F1

If file 1 is not open, the program loops back to program level 1 to open the necessary files.

\$DELETE FORMAT 203

\$FORMAT 203 /* PROGRAM LEVEL 3--CLOSES FILES AND ENDS PROCESSING */

@3,1 E3,80

Move the cursor to row 3 column 1. Erase the screen to row 3 column 80.

SET R2(10,22)="TOTAL RECORDS DELETED" SET R2(33,12)=A1

Place "TOTAL RECORDS DELETED" and the contents of accumulator 1 in file 2's record buffer.

WRITE2,S2

Skip two lines, then print file 2's record buffer.

CLOSE1 CLOSE2

Close both files.

NDR P(H)"PROGRAM COMPLETE" G(L)1

Display a prompt to inform the operator that processing is finished, with a one-character field to keep the prompt on the screen.

EXAMPLE 2: REFORMATTING DATA AS IT IS ENTERED

The program on the following page reformats data to a batch in a different job as an operator is keying in the data.

In this example:

- The job ENTRYJOB provides the operator interface. ENTRYJOB is not an MSFP file. It prompts the operator and keeps a record count.
- Program level 1 of ETNRYJOB (format 401) initiates the program.
- Program level 2 (format 402) prompts the operator to enter data.
- Program level 3 (format 403) completes the program.
- Accumulator 1 counts the number of records entered.

Appendix A
Sample Programs

- RFMTJOB is the job into which data from ENTRYJOB is to be reformatted. It has one program level (format 400). RFMTJOB is defined as Ftype 400 (an SD file), with the batch name to be entered at runtime. Ftype 400 is logical file 1.

Example 2:

```

$DEFJOB ENTRYJOB   $FORMAT 401,402,403   $MAXSIZ 140 $ACCUMS 1
$DEFJOB RFMTJOB    $FORMAT 400           $MAXSIZ 140 $ACCUMS 1
$FTYPE 400,SD      $JOB RFMTJOB,*

$DELETE FORMAT 401
$FORMAT 401          /* PROGRAM LEVEL 1--STARTS DATA COLLECTION */
ASD NDR B4,20 P(H)"ENTERING INFORMATION FOR JOB RFMTJOB"
E22,80
OPEN1,400,0
B6,25 P(H)"TYPE Y TO BEGIN PROCESSING"
B7,25 P(H)"OTHERWISE, PRESS EXIT KEY" B3 1A(L)1E="Y"
IF OPENED1
THEN (B23,1 G(L)1S Z1 F2)
ELSE (P(H)"OUTPUT BATCH NOT OPENED " B23,1 G(L)1)

$DELETE FORMAT 402
$FORMAT 402          /* PROGRAM LEVEL 2--COLLECTS AND PROCESSES DATA */
E22,80
P(H)2,1"NAME"        P(H)3,1"ADDRESS"
P(H)4,1"CITY"        P(H)5,1"STATE"
P(H)6,1"INVOICE #"   P(H)7,1"TOTAL DUE"
BLANKR1
@ (H)2,12 1G(L)30 SET R1(15,30)=F1
@ (H)3,12 2G(L)30 SET R1(45,30)=F2
@ (H)4,12 3G(L)30 SET R1(75,30)=F3
@ (H)5,12 4G(L)10 SET R1(105,10)=F4
@ (H)6,12 5I(L)6 SET R1(1,6)=F5
@ (H)7,12 6L(L)8.2 SET R1(7,8)=F6
WRITE1
SET A1=A1+1
B21,1
P(H)"PRESS SPACEBAR TO CONTINUE OR TYPE Y IF FINISHED WITH BATCH"
A(L)1E="SA" Y" IF="Y" THEN F3 ELSE F2

$FORMAT 403 /* PROGRAM LEVEL 3--ENDS PROCESSING */
E22,80 NDR B4,1 P(H)"TOTAL RECORDS WRITTEN " G(L)6GA1
B6,1 P(H)"PROGRAM COMPLETE; PRESS EXIT " CLOSE1 G(L)1

$FORMAT 400

E22,80

B3,1 P(H)"INVOICE " I(L)6
B4,1 P(H)"TOTAL DUE " L(L)8.2
B5,1 P(H)"NAME " G(L)30
B6,1 P(H)"ADDRESS " G(L)30
B7,1 P(H)"CITY " G(L)30
B8,1 P(H)"STATE " G(L)10

```

Appendix A
Sample Programs

Explanation of Example 2:

\$DEFJOB ENTRYJOB \$FORMAT 401,402,403 \$MAXSIZ 140 \$ACCUMS 1
Define the job ENTRYJOB.

\$DEFJOB RFMTJOB \$FORMAT 400 \$MAXSIZ 140 \$ACCUMS 1
Define the job RFMTJOB.

\$FTYPE 400,SD \$JOB RFMTJOB,*
Create an SD file type definition. Assign the identifier 400 to the SD file. The file type definition associates Ftype 400 with the job RFMTJOB; the batch name is to be entered at run time.

\$DELETE FORMAT 401
\$FORMAT 401 /* PROGRAM LEVEL 1 */
Start format 401 (program level 1 for ENTRYJOB).

ASD NDR B4,20 P(H)"ENTERING INFORMATION FOR JOB RFMTJOB"
Turn on Auto Skip Dup and specify that the record is not written to disk (NDR). Blank to row 4, column 20, and put a prompt on the screen.

E22,80
Blank the rest of the screen.

OPEN1,400,0
Assign Ftype 400 as logical file 1 and open the file for output. When this command executes, the operator is prompted to enter a name for the output batch. The prompt and cursor are on the message line.

B6,25 P(H)"TYPE Y TO BEGIN PROCESSING"
B7,25 P(H)"OTHERWISE, PRESS EXIT KEY" B3 1A(L)1E="Y"
Prompt the operator to begin by pressing Y or to stop by pressing EXIT.

IF OPENED1 THEN (B23,1 G(L)1S Z1 F2)
Test to determine if file 400 is open. If it is, zero accumulator 1 and go to program level 2.

ELSE (P(H)"OUTPUT BATCH NOT OPENED " B23,1 G(L)1)
If the output batch isn't opened, notify the operator.

\$DELETE FORMAT 402
\$FORMAT 402 /* PROGRAM LEVEL 2 */
Program level 2 puts prompts on the screen.

E22,80
P(H)2,1"NAME" P(H)3,1"ADDRESS"
P(H)4,1"CITY" P(H)5,1"STATE"
P(H)6,1"INVOICE #" P(H)7,1"TOTAL DUE"
Blank the screen, then display the prompts before the operator enters any data.

BLANKR1

Blank file 1's record buffer.

```
@(H)2,12  1G(L)30    SET R1(15,30)=F1
@(H)3,12  2G(L)30    SET R1(45,30)=F2
@(H)4,12  3G(L)30    SET R1(75,30)=F3
@(H)5,12  4G(L)10    SET R1(105,10)=F4
@(H)6,12  5I(L)6     SET R1(1,6)=F5
@(H)7,12  6L(L)8.2   SET R1(7,8)=F6
```

As each field is entered, copy the data into the record buffer, placing the invoice number and total due before the customer's name and address.

WRITE1

Write the contents of the record buffer into the output batch.

SET A1=A1+1

Increment accumulator 1 by one to keep a record count.

B21,1

P(H)"PRESS SPACEBAR TO CONTINUE OR TYPE Y IF FINISHED WITH BATCH"

A(L)1E=SA" Y" IF="Y" THEN F3 ELSE F2

Display a prompt to ask whether the batch is finished. If the batch is not finished, repeat program level 3. If the batch is finished, go to program level 2.

\$FORMAT 403 /* PROGRAM LEVEL 3 */

Program level 3 closes file 1 and ends processing.

E22,80 NDR B4,1 P(H)"TOTAL RECORDS WRITTEN " G(L)6GA1

Erase the screen and display the total number of records written (from accumulator 1).

B6,1 P(H)"PROGRAM COMPLETE; PRESS EXIT " CLOSE1 G(L)1

Prompt the operator to exit the batch, then close file 1.

\$FORMAT 400

Start format 400 (program level 1 of RFMTJOB).

E22,80

Blank the screen.

```
B3,1 P(H)"INVOICE " I(L)6
B4,1 P(H)"TOTAL DUE " L(L)8.2
B5,1 P(H)"NAME " G(L)30
B6,1 P(H)"ADDRESS " G(L)30
B7,1 P(H)"CITY " G(L)30
B8,1 P(H)"STATE " G(L)10
```

Display prompts.

THE [illegible] OF [illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



Appendix B MSFP Printing with lp

This section describes how to do MFSP printing if your LiFE-Works system is using the traditional operating system (lp) facility instead of the Motorola Print System. If your system does use the Motorola Print System, use the instructions in Section 8, "MFSP Printing," instead.

ABOUT MSFP PRINTING

MSFP lets you do several print operations that can't be done through other kinds of LiFE-Works printing. When you print through MSFP you can:

- Access a printer directly from a format program.
- Select and print individual records from a batch. (You can select and print individual index set records using either MSFP or non-MSFP methods.)
- Precisely control the vertical movement of paper through the printer.

This section describes printing procedures that are specific to MSFP. This section assumes that you are familiar with LiFE-Works printing as described in the LiFE-Works Supervisor's Manual.

This section covers the following topics:

- Creating Vertical Format Unit (VFU) definitions. LiFE-Works uses VFU simulation to move the paper through the printer.
- Defining MSFP forms. An MSFP form definition lets you define a specific printer/form combination as a PR file and refer to it in an MSFP program.
- Using spool files. MSFP prints by writing data to a spool file, then sending the spool file to the printer. You can choose between permanent or temporary spool files, and you can name the spool file yourself or have the system generate a name.
- Using the Spool File Display (mode P-S). The Spool File Display lists permanent spool files and lets you print or delete listed files.

CREATING A VERTICAL FORMAT UNIT (VFU) DEFINITION

A vertical format unit (VFU) is a device that controls the vertical movement of paper through a printer. LIfE-Works does VFU simulation; that is, it uses software instructions to accomplish the functions of a physical VFU device.

You don't need to know much about physical VFU devices to do MSFP printing. However, you do need to understand the basic concepts and terminology. If you are not familiar with VFU devices, read "About VFU's" on the next page. If you are familiar with VFU's, just keep in mind that for LIfE-Works, "channels," "punches," and so on refer to software instructions rather than parts of a device.

All LIfE-Works printing uses VFU simulation for printer control. However, non-MSFP printing methods use only carriage control channels 1 and 2. (Typically, channel 1 makes the paper advance to top-of-form, and channel 2 makes the paper advance one line at a time.) MSFP lets you take full advantage of a VFU. Using the MSFP WRITEfn command, you can:

- Access any of the 12 VFU channels (WRITEfn,Cc).
- Make the paper advance a specified number of lines (WRITEfn,Ss).
- Prevent the paper from moving, thus causing overprinting (WRITEfn,CR).

A standard VFU, VFU 000, is already on your system. You can use VFU 000 in programs, but you cannot delete or modify it. You can also create other VFU definitions as necessary.

To create a VFU definition:

- 1 Access the LWconfig program using whatever procedures apply to your system.
- 2 Select "PRINTING" from the LIfE-Works CONFIGURATION PROGRAM menu.
- 3 Select "VFU DEFINITIONS" from the ALTER PRINT SYSTEM CONFIGURATION menu.
- 4 Select "DEFINE" from the ALTER VFU DEFINITIONS menu. This menu lists existing VFU definitions.
- 5 Enter a unique three-digit identifier (001-999) for the new VFU, then press TAB.
- 6 Length of VFU in lines is the total number of lines on the page, with no top or bottom margin allowance. The default value, 066, is the length of an 11-inch page at 6 lines per inch. To keep the default value, press TAB. To change the value, type a three-digit number (001-122) over the default value, then press TAB.

About VFU's

A VFU controls paper movement through a printer by means of a punched paper tape. The tape is called a VFU tape or carriage control tape.

A carriage control tape is usually 12 channels (columns) wide and as many lines long as one page of the paper it is to be used with. The carriage control tape is a loop; the ends are connected so that the first line on the tape follows the last line. As each page moves through the printer, the tape moves with it, line for line.

The controlling application--whatever software is printing--alternately sends data to the printer and instructs the printer to skip to a specified channel on the tape. When the printer receives a skip-to-channel-x instruction, it advances the paper until it encounters the next "punch" (hole) in channel x.

7 Number of print lines is the total number of lines on the page minus the top and bottom margin allowance. The default value of 060 creates a three-line top margin and a three-line bottom margin. To keep the default value, press ACCEPT. To change the value, type a three-digit number (001-122) over the default value, then press ACCEPT. The value you enter must be equal to or less than the total number of lines on the page (item 6 above).

8 A prompt for channel definition 1 appears. Do one of the following:

- Type another channel number (02-12) over the 01, then press ACCEPT.
- or
- Press ACCEPT as many times as necessary to reach the prompt for the channel you want to define.

NOTE

If you change channels 1 and 2 of a VFU, be aware that your changes will affect any LiFE-Works printing that uses that VFU, not just MSFP printing.

9 For each channel, "punch" the channel as desired by typing line numbers in the space provided. Separate the numbers with commas. For example, to punch lines 1, 30, and 60 of channel 4, enter:

Definition for channel #: 04

1,30,60

You can also use the following abbreviations in channel definitions:

- FF (form feed) puts a punch in line 1 only.
- LF (line feed) puts punches in lines 1-60.

If you use either of these abbreviations, the abbreviation must be the only thing in the channel definition.

- 10 When you finish the channel definitions, press EXIT to exit, then press ACCEPT to save the new VFU definition.

NOTE

The ALTER VFU DEFINITIONS menu also lets you view, modify, or delete VFU definitions. Follow the screen prompts for these operations. You can get a help display at any point by pressing the HELP key.

DEFINING MSFP FORMS

An MSFP form lets you include the following information as part of an MSFP program:

- The logical printer to be used
- The LWconfig form to be used
- A heading to be printed at the top of each page

To define an MSFP form, enter and execute the following supervisory commands:

```
$FORMS vvv  
  
[F=form] [P=printer] [H=heading]
```

where

vvv is a unique three-digit identifier (000-999).

printer is the name of the logical printer to be used. If you omit the P parameter, MSFP uses whatever printer is currently assigned to the terminal running the MSFP format program.

form is the LWconfig printer form to be used. If you omit the F parameter, MSFP uses whatever form is currently assigned to the terminal running the MSFP format program.

heading generates a heading at the top of each page. Use the following commands to generate a heading:

Tw,text generates a text string w characters long, where text is the text string.

Sw skips (blanks) w columns.

Gw,c generates w repetitions of the character c. c can be any character except a blank.

P generates sequential page numbers. Page numbers start at 1 and can go up to 99999. Page numbers are right-justified within the possible five character positions.

#keyword generates the current value of a system keyword. keyword can be any LIFE-Works system keyword except those used specifically with MSFP.

/ causes a line feed.

The heading can contain up to a total of 190 characters, including skipped columns and line feeds.

Usage Notes

- To use an MSFP form in a program, you refer to the form in the \$FORMS vvv parameter of a PR file type definition (where vvv is the three-digit MSFP form identifier). If you don't specify an MSFP form, print output from your program is sent to whatever printer/form is currently assigned to the terminal running the MSFP format program.
- MSFP forms are system elements, like formats and value sets. They can be used by any number of MSFP programs. Select mode S-D to see a list of the MSFP forms on your system.
- \$FORMS vvv must be on a record by itself. The remaining \$FORMS parameters can extend over multiple records (but individual parameters can't be split).
- You can use separating commas rather than blanks between the \$FORMS parameters. (See "Examples" below.)
- Generating a heading in an MSFP form is much like generating a heading in disk-to-printer reformatting. If you're not familiar with this procedure, see the LiFE-Works Supervisor's Manual.
- For information on defining LWconfig forms, see the LiFE-Works Supervisor's Manual.

Examples

```
$FORMS 111  
F=INVOICE P=CP1-INVOICE
```

Defines logical printer CP1-INVOICE and LWconfig form INVOICE as MSFP form 111.

```
$FORMS 200  
F=REPORT,P=CP2-REPORT  
H=S20 T48,COPY CENTER CROSS-CHARGES (SORTED BY DEPARTMENT) S5 PAGE: P
```

Defines logical printer CP2-REPORT and LWconfig form REPORT as MSFP form 200, generates a page heading, and numbers the pages sequentially.

USING SPOOL FILES

Each MSFP PR file (printer) has a spool file associated with it. When a format program writes data to a PR file, the spool file receives the data. When the program closes the PR file, the associated spool file is placed in the print queue.

For each PR file, you can specify either a permanent spool file or a temporary spool file.

Permanent spool files remain on the system after being printed. You don't have to print a permanent spool file when you close it; you can delay printing by closing the file with the delay option (CLOSEfn,D).

Temporary spool files are deleted after printing. You cannot delay printing of a temporary spool file.

Use the S option of the OPENfn command to specify a permanent spool file, or use the T option to specify a temporary spool file. If you don't specify either S or T, MSFP uses a temporary spool file with a system-generated name.

NOTE

If you write a program that opens and closes the same PR file multiple times, your choice of spool files affects the way output data is handled. If you use a permanent spool file, the system appends output data to the end of the spool file each time you close the PR file. (Page numbering doesn't start over with each addition.) If you use a temporary spool file, the system sends the data to a new spool file each time you close the PR file.

You have a number of choices for naming spool files. You can:

- Supply a spool file name in the \$SPOOL parameter of the \$FTYPE or FTYPEfn command:

\$FTYPE...\$SPOOL spoolfile

- Supply a spool file name in the OPENfn command:

$$\text{OPENfn} \dots \left\{ \begin{array}{c} \text{S} \\ \text{T} \end{array} \right\} = \text{"spoolfile"}$$

- Generate a spool file name into the OPENfn command from another source, such as an accumulator or buffer:

$$\text{OPENfn} \dots \left\{ \begin{array}{c} \text{S} \\ \text{T} \end{array} \right\} = \text{reference}$$

- Have the spool file name entered at run time:

\$FTYPE ...\$SPOOL *

- Have the system generate the spool file name:

$$\text{OPENfn} \dots \left\{ \begin{array}{c} \text{S} \\ \text{T} \end{array} \right\} \quad \text{or} \quad \text{OPENfn with no spool file option--generates a temporary spool file}$$

Spool file names can be 1-14 characters long. They can contain any characters except slashes (/) and backslashes (\), but the first character must not be a period.

System-generated permanent spool file names have the form

Vtttffnnn

where

ttt is the current terminal number.

ff is the logical file number assigned to the PR file.

nnn is the form number if \$FORMS was included in the \$FTYPE or FTYPEfn command. If not, nnn is blank.

System-generated temporary spool file names have the form

LWSPxxxxxx

where xxxxxx is six random characters.

If you use more than one naming method in the same program, LiFE-Works applies the following precedence:

1. The name entered at run time (highest precedence).
2. The name specified in (or generated into) the OPENfn command.
3. The name specified in the \$FTYPE or FTYPEfn command.
4. The system-generated name (lowest precedence).

USING THE SPOOL FILE DISPLAY (MODE P-S)

Select Mode P-S to see a list of all spool files currently on the system. These files are usually permanent spool files, but temporary spool files also appear in the mode P-S display while they are receiving print data.

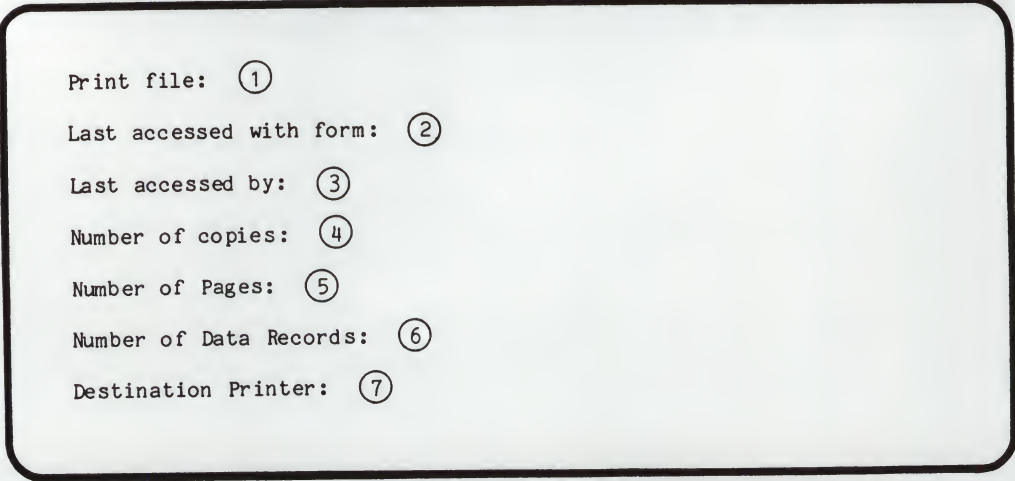
Mode P-S also lets you:

- Get detailed information about a permanent spool file.
- Print a permanent spool file.
- Delete a permanent spool file.
- Print a permanent spool file, then delete it.

NOTE

Use mode P-S to print a permanent spool file closed with the delay option (CLOSEfn,D).

To view information about a permanent spool file, select mode P-S, then enter the spool file name as prompted. The following display appears (Figure 8-1):



Print file: ①
Last accessed with form: ②
Last accessed by: ③
Number of copies: ④
Number of Pages: ⑤
Number of Data Records: ⑥
Destination Printer: ⑦

Figure B-1. Mode P-S Display for an Individual Spool File

Explanation:

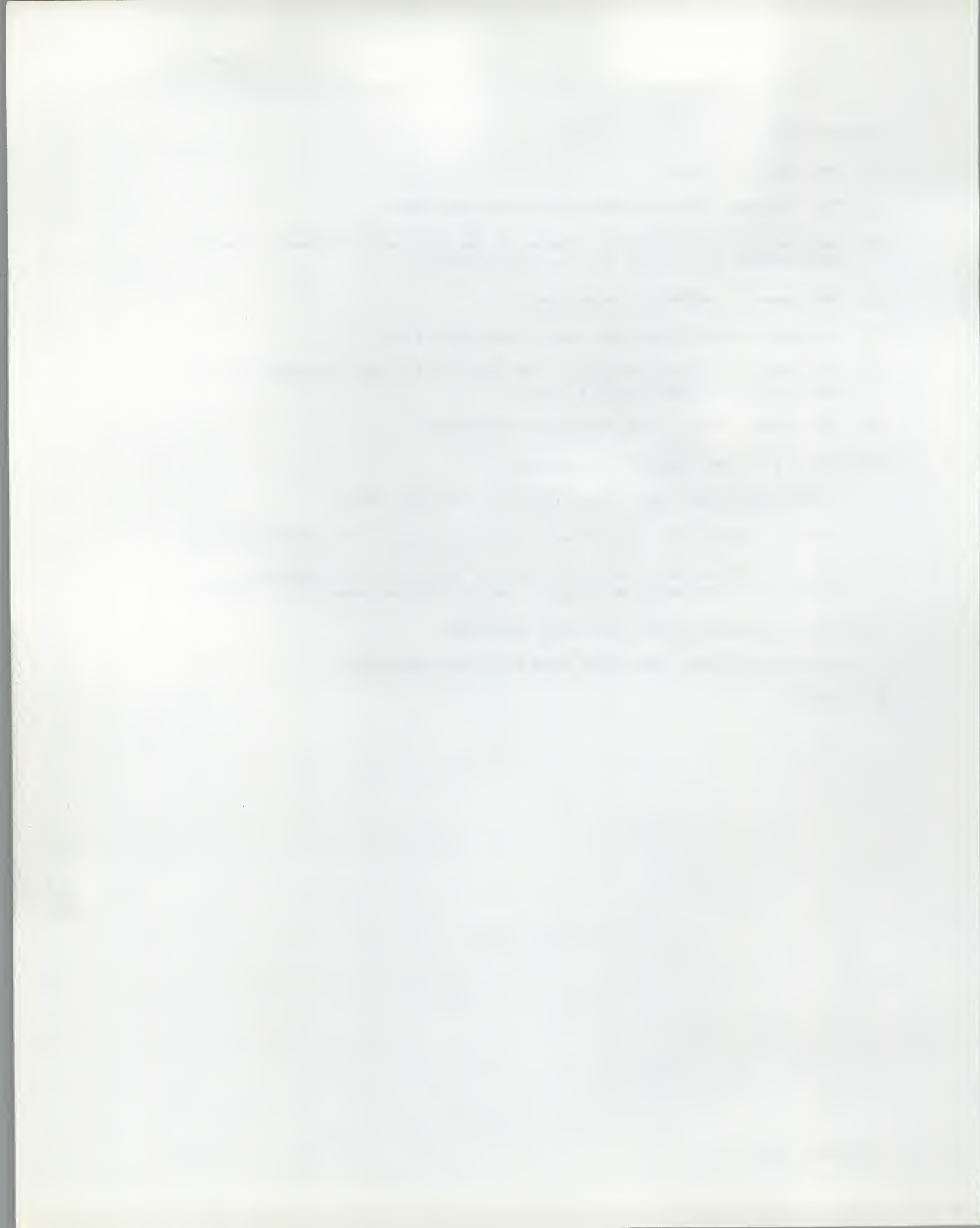
- ① The spool file name.
- ② The form used the last time the file was written to.
- ③ The operator ID and terminal number of the last person to modify the file, and the date and time of the last modification.
- ④ The number of copies to be printed.
- ⑤ The page number of the last page in the print file.
- ⑥ The number of records written to the print file. (Page headings and blank lines aren't included in this count.)
- ⑦ The printer on which the file is to be printed.

To print a permanent spool file using mode P-S:

- 1 Get the individual spool file display as described above.
- 2 Press P to print the spool file, or press Q to print and delete the file.
- 3 Enter the name of the printer you want to use, then press RETURN; or press RETURN without entering anything to use the printer named in the display.

To delete a permanent spool file using mode P-S:

- 1 Get the individual spool file display as described above.
- 2 Press D.





INDEX

- [symbol in command strings, 1-3
- @ sign in OPENfn command, 4-13
- { symbol in command strings, 1-3

- Alternate choices in command strings, 1-3
- An descriptor
 - Generating a file type definition from the, 3-15
 - Generating LOCATEfn command parameters from the, 4-9
 - Generating OPENfn command parameters from the, 4-14
 - Generating READfn(KEY...) command parameters from the, 4-20
- AS file
 - Defining an, 3-10
 - Opening an, 4-12, 4-17
 - System keyword values set by opening, 4-17

- B(c,l) descriptor
 - Generating a file type definition from the, 3-15
 - Generating LOCATEfn command parameters from the, 4-9
 - Generating OPENfn command parameters from the, 4-14
 - Generating READfn(KEY...) command parameters from the, 4-21
- Batch name, generating or validating, 7-2
- #BATCHfn system keyword, 7-2
 - And the OPENfn command, 4-16
- Blanking the record buffer, 5-2
- BLANKRfn command, 5-2
- BOBfn conditional indicator, 6-2
 - And the LOCATEfn command, 4-10
 - And the OPENfn command, 4-15, 4-16
- BOF parameter of the LOCATEfn command, 4-9
- BOFfn conditional indicator, 6-3
 - And the LOCATEfn command, 4-9
 - And the OPENfn command, 4-15, 4-16
- Buffers, use of, 2-4

- C parameter of the CLOSEfn command, 4-3
- C parameter of the OPENfn command, 4-13
- Carriage control channels
 - Defining, B-3, B-4
 - And WRITEfn command, 4-29
- Carriage return (line feed), suppressing (WRITEfn,CR), 4-29
- Cc parameter of the WRITEfn command, 4-29
- Clearing the records from an SD file (OPENfn...C), 4-13
- CLOSEfn command, 4-3
 - Delay option (CLOSEfn,D), 4-3, 8-5, B-7
- Commands, general
 - CLOSEfn, 4-3
 - LOCATEfn, 4-9
 - OPENfn, 4-12
 - READfn, 4-20

Index

- REWRITEfn, 4-27
- WRITEfn, 4-29
- Summary of, 4-1
- Commands used with index sets
 - DELETEDfn, 4-5
 - INSERTfn, 4-7
 - READfn, 4-20
 - READfn(KEY=...), 4-20
 - READfn(KEY>=...), 4-20
 - RELEASEfn, 4-26
- COMP flag, setting (CLOSEfn,C), 4-9
- Conditional indicators
 - BOBfn, 6-2
 - BOFfn, 6-3
 - EOBfn, 6-4
 - EOFFfn, 6-5
 - OPENEDfn, 6-6
 - OPENINfn, 6-7
 - OPENOUTfn, 6-8
 - RECSELfn, 6-9
 - Summary of, 6-1
- CR parameter of the WRITEfn command, 4-29
- D parameter of the CLOSEfn command, 4-3, 8-5, B-7
- DELETEDfn command, 4-5
- EOBfn conditional indicator, 6-4
 - And the LOCATEfn command, 4-10
 - And the OPENfn command, 4-15, 4-16
- EOF parameter of the LOCATEfn command, 4-9
- EOFFfn conditional indicator, 6-5
 - And the LOCATEfn command, 4-10
 - And the OPENfn command, 4-15, 4-16
- #ERRORfn system keyword, 7-3
- File permissions, 3-11
- File type definitions
 - For AS files, 3-10
 - For PR files, 3-8
 - For SD files, 3-3
 - For XD files, 3-6
 - For UX files, 3-14
- #FILEfn system keyword, 7-5
 - And OPENfn command, 4-16, 4-17
- Flags that prevent opening, 4-16
- Fn descriptor
 - Generating a file type definition from the, 3-15
 - Generating LOCATEfn command parameters from the, 4-9
 - Generating OPENfn command parameters from the, 4-14
 - Generating READfn(KEY...) command parameters from the, 4-21
- Form, MSFP, 8-3, 8-4, B-5, B6
- \$FORMS command, 8-3, 8-4, B-5, B-6

- \$FORMS parameter of \$FTYPE or FTYPEfn command, 3-8
- \$FTYPE supervisory command, 3-1
 - And AS files, 3-10
 - And PR file type, 3-8
 - And SD file type, 3-3
 - And XD file type, 3-6
 - And UX file type, 3-14
- Ftype system element
 - Defined, 3-1
 - Ftype directories (modes S-T and S-U), 3-1
- FTYPEfn format command, 3-1, 3-15
 - And AS files, 3-10
 - And PR file type, 3-8
 - And SD file type, 3-2
 - And XD file type, 3-6
 - And UX file type, 3-14
- I parameter of the OPENfn command, 4-13
- Index sets (see Commands used with index sets)
- \$INDSET command, 4-17
- \$INDSET parameter of \$FTYPE or FTYPEfn command, 3-6
- Input, opening a file for, 4-15
- INSERTfn command, 4-7
- Job name, generating or validating, 7-6
- #JOBfn system keyword, 7-6
 - And OPENfn command, 4-16, 4-17
- Key field name, specifying for index sets, 4-20
- Key parameter of READfn command, 4-20, 4-23
- L parameter of the CLOSEfn command, 4-3
- LAN (see OfficeLAN)
- \$LENGTH parameter of \$FTYPE or FTYPEfn command, 3-8
- LFn descriptor
 - Generating a file type definition from the, 3-15
 - Generating LOCATEfn command parameters from the, 4-9
 - Generating OPENfn command parameters from the, 4-14
 - Generating READfn(KEY...) command parameters from the, 4-21
- Local Area Network (see OfficeLAN)
- LOCATEfn command, 4-9
- LOCK flag, setting (CLOSEfn,L), 4-3
- Loop concept, 2-3
- \$MAXSIZE parameter for defining an AS file, 3-10
 - Mode P-S, 8-8, 8-11, B-10, B-11
 - Mode S-T, 3-1
 - Mode S-U, 3-1
 - MSFP form, 8-3, B-5
- NC parameter of the OPENfn command, 4-13
- \$NODE parameter of the \$FTYPE command, 3-3, 3-4, 3-6

Index

- #NODEfn system keyword, 7-7
- Number of copies to be printed (OPENfn...NC), 4-13
- NW parameter of the READfn command, 4-21, 4-23

- O parameter of the OPENfn command, 4-13
- OfficeLAN
 - #ERRORfn system keyword, 7-3
 - \$NODE parameter of the \$FTYPE command, 3-3, 3-4, 3-6
 - #NODEfn descriptor, 7-7
 - RESULT and RESULT# commands, 7-3, 7-4
- OPENEDfn conditional indicator, 6-6
 - And the CLOSEfn command, 4-3
 - And the OPENfn command, 4-15
- OPENfn command, 4-12
- OPENINfn conditional indicator, 6-7
 - And the CLOSEfn command, 4-3
 - And the OPENfn command, 4-15
- OPENOUTfn conditional indicator, 6-8
 - And the CLOSEfn command, 4-3
 - And the OPENfn command, 4-15
- Optional parameters in command strings, 1-3
- Order chain, 2-5, 3-3, 3-4
 - And the BOBfn conditional indicator, 6-2
 - And the BOFfn conditional indicator, 6-3
 - And the EOBfn conditional indicator, 6-4
 - And the EOFfn conditional indicator, 6-5
 - And the LOCATEfn command, 4-10
- \$ORDER parameter of \$FTYPE or FTYPEfn command, 3-3, 3-4
- Output, opening a file for, 4-15

- P parameter of the OPENfn command, 4-12, 4-13
- Page heading in MSFP form, B-5, B-6, 8-3, 8-4
- Password-protected job, opening a, 4-16
- \$PATH parameter for defining an AS file, 3-10
- Permanent spool file, 4-13, 8-7
- Permissions, 3-11, 4-13
- #PLEVELfn system keyword, 7-8
 - And the OPENfn command, 4-15
 - And the READfn command, 4-22
 - And the WRITEfn command, 4-30
- PR file
 - Defining a, 3-8
 - Opening a, 4-12, 4-17
 - System keyword values set by opening a, 4-15 to 4-17
- Printing through MSFP
 - Creating VFU definitions, 8-2, B-2
 - Defining MSFP forms, 8-3, B-5
- Spool files
 - General information and naming conventions, 8-5, B-7
 - Printing and deleting permanent spool files, 8-8, B-10
- Processing loop, 2-3
- Program level, generating, validating, or setting, 7-8

- R parameter of the READfn command, 4-20
- R parameter of the REWRITEfn command, 4-27
- R parameter of the WRITEfn command, 4-29
- READfn command, 4-20
 - And index sets, 4-22
 - And the LOCATEfn command, 4-10
- READfn(KEY=...) command, 4-20, 4-23
- READfn(KEY>=...) command, 4-20, 4-23
- RECLFN parameter of the READfn command, 4-21, 4-23
- #RECLFN system keyword, 7-9
- #RECFN system keyword, 7-11
- Record buffer
 - Accessing with the Rfn command, 5-3
 - Blanking the (BLANKRfn), 5-2
 - In MSFP programming, 2-4
 - Summary of commands used with the, 5-1
- Record length, generating, validating, or setting, 7-9
- Record number, generating or validating, 7-11
- Record pointer
 - Defined, 1-4
 - And the LOCATEfn command, 4-10
- RECSELfn conditional indicator, 6-9
- RELEASEfn command, 4-26
- Releasing access to a record in an XD file
 - RELEASEfn, 4-26
 - REWRITEfn,R, 4-27
 - WRITEfn,R, 4-29
- RESULT and RESULT# commands, 7-3, 7-4
- Rewinding a file (READfn,R), 4-20
- REWRITEfn command, 4-27
- Rfn(c,l) descriptor
 - Generating a file type definition from the, 3-15
 - Generating LOCATEfn command parameters from the, 4-9
 - Generating OPENfn command parameters from the, 4-14
 - Generating READfn(KEY...) command parameters from the, 4-21
 - And the SET command, 5-3
 - Using Rfn to access record buffer, 5-3
- RS(c,l) descriptor
 - Generating a file type definition from the, 3-15
 - Generating LOCATEfn command parameters from the, 4-9
 - Generating OPENfn command parameters from the, 4-14
 - Generating READfn(KEY...) command parameters from the, 4-21
- S parameter of the OPENfn command, 4-13
- S(r,c,l) descriptor
 - Generating a file type definition from the, 3-15
 - Generating LOCATEfn command parameters from the, 4-9
 - Generating OPENfn command parameters from the, 4-14
 - Generating READfn(KEY...) command parameters from the, 4-21
- SD file
 - Defining an, 3-3
 - Opening an, 4-12
 - System keyword values set by opening an, 4-15, 4-16

Index

SET statement

- Rfn descriptor and, 5-3

- System keywords and, 7-1 to 7-10

- Skipping print lines (WRITEfn,Ss), 4-29

Spool file

- And the CLOSEfn,D command, 4-3, 8-5, B-7

- Naming, 8-5, B-7

- Permanent, 4-13, 8-5, B-7

- Temporary, 4-13, 8-5, B-7

- Using mode P-S to print or delete, 8-8, 8-11, B-10, B-11

- Spool file name, generating or validating, 7-5

- \$SPOOL parameter of \$FTYPE or FTYPEfn command, 3-8, 3-9

- Ss parameter of the WRITEfn command, 4-29, 4-30

- \$STTY parameter for defining an AS file, 3-10

.symbol. descriptor

- Generating a file type definition from the, 3-15

- Generating LOCATEfn command parameters from the, 4-9

- Generating OPENfn command parameters from the, 4-14

- Generating READfn(KEY...) command parameters from the, 4-21

System keywords, 7-1

- #BATCHfn, 7-2

- #ERRORfn, 7-3

- #FILEfn, 7-5

- #JOBfn, 7-6

- #NODEfn, 7-7

- #PLEVELfn, 7-8

- #RECLFNfn, 7-9

- #RECNUMfn, 7-11

- T parameter of the OPENfn command, 4-13

- Temporary spool file, 4-13, 8-7

- \$TIMEOUT parameter for defining an AS file, 3-10

- TO parameter of the OPENfn command, 4-12, 4-13

- UNTIL parameter of the READfn command, 4-21, 4-23

UX file

- Defining a, 3-13

- Opening a, 4-12

- System keyword values set by opening, 4-18

Validation/generation descriptors, 7-1

- #BATCHfn, 7-2

- #ERRORfn, 7-3 to 7-4

- #FILEfn, 7-5

- #JOBfn, 7-6

- #NODEfn, 7-7

- #PLEVELfn, 7-8

- #RECLFNfn, 7-9

- #RECNUMfn, 7-11

- Vertical Format Unit (VFU) definition, 8-2, B-2

- W parameter of the READfn command, 4-21, 4-23

- \$WAIT parameter for defining an AS file, 3-10
- Work buffers, Using in MSFP programming, 2-4
- WRITEfn command, 4-29
 - And the LOCATEfn command, 4-10
- X(iii,c,1) descriptor
 - Generating a file type definition from the, 3-15
 - Generating LOCATEfn command parameters from, 4-9
 - Generating OPENfn command parameters from the, 4-14
 - Generating READfn(KEY...) command parameters from the, 4-21
- XD file
 - Creating a file type definition for an, 3-6
 - Opening an, 4-12
 - System keyword values set by opening an, 4-15, 4-17

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO

THE UNIVERSITY OF CHICAGO





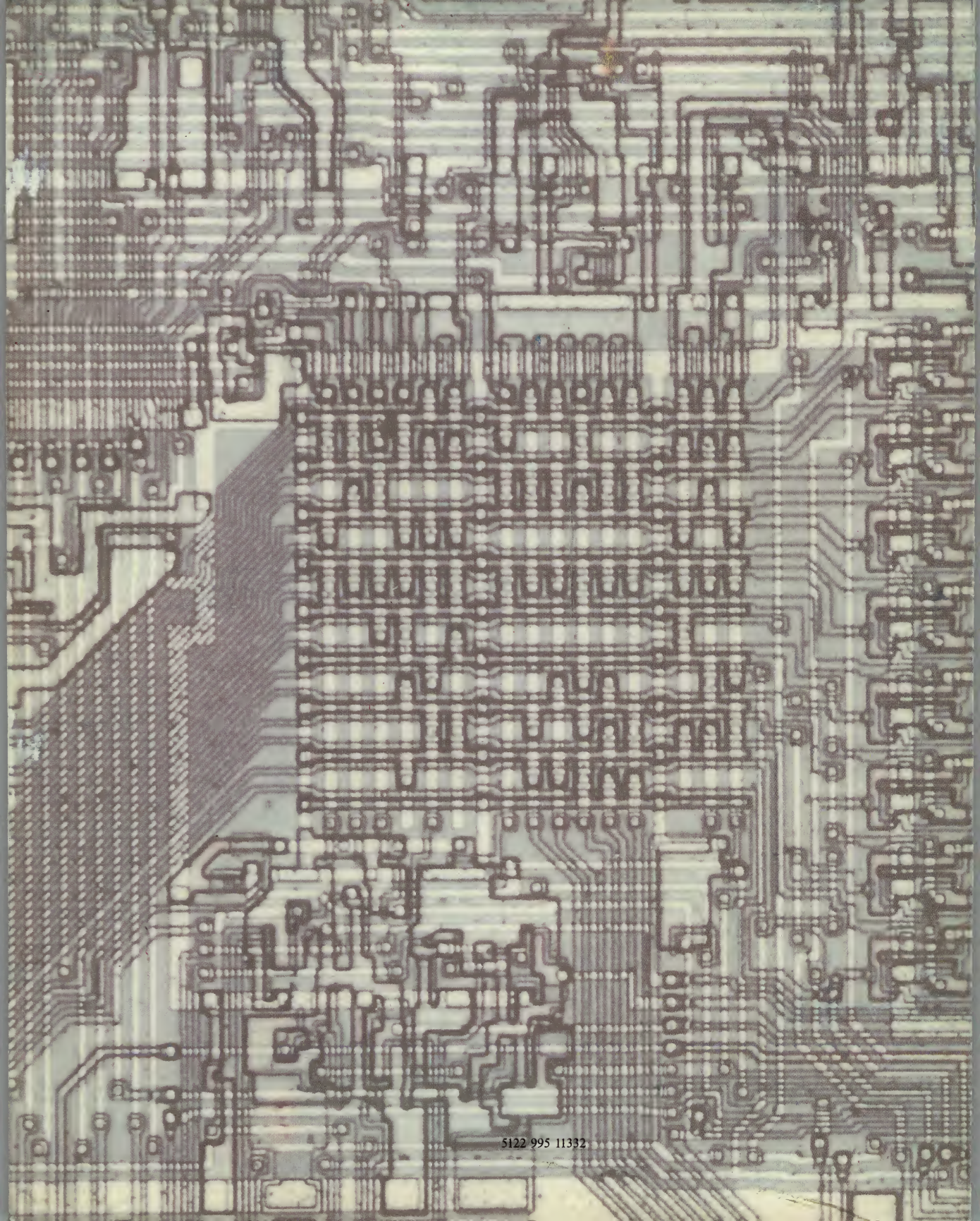












5122 995 11332